RL-TR-94-216, Volume I (of two)
Final Technical Report
December 1994

NEURAL NETWORK FALSE ALARM FILTER

Raytheon Company

F. Aylstock, L. Elerin, J. Hintz, C. Learoyd, and R. Press

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

This report documents the Neural Network False Alarm Filter (NNFAF) contract, an investigation and demonstration of the use of neural network technology to improve Built-in Test (BIT) by filtering out false alarms from the BIT output and by identifying intermittent faults.

Neural Network False Alarm Filter Project Description

The NNFAF contract was performed by Raytheon Company for Rome Laboratory at Griffiss Air Force Base, New York. It was a 2-year research contract which began in September, 1992 and ended in September, 1994. The purpose of the project was to identify, develop, and demonstrate a set of approaches for applying neural network learning techniques to improve the performance of BIT. The approaches focused on the need to filter out false alarms and to identify intermittent failures.

The NNFAF methodology involved a state-of-the-art assessment of neural networks and BIT techniques, the selection of candidate BIT technique and neural network combinations for analysis, simulation of BIT techniques to generate neural network training and testing data, neural network development and training, and results analysis. The selected approaches were developed and implemented in a prototype, proof-of-principle demonstration. In addition, the impact of their implementation in fielded avionics systems was examined and the potential costs and benefits were analyzed.

Many modern systems that employ BIT to automatically detect and report failures are susceptible to reporting a failure when the system is actually functional (false alarm). As a result, functional systems are often taken off-line for repair, where they are re-tested as functional (Re-test OK). Consequently, system availability is reduced and maintenance costs are increased due to increasing repair actions. Simple filtering techniques can be applied to limit the number of false alarms that are reported. However, the risk of filtering out valid failure reports increases with the amount of false alarm filtering. The NNFAF project investigated the application of neural networks (NNs) to BIT for the purpose of classifying BIT reports as false alarms or valid failures.

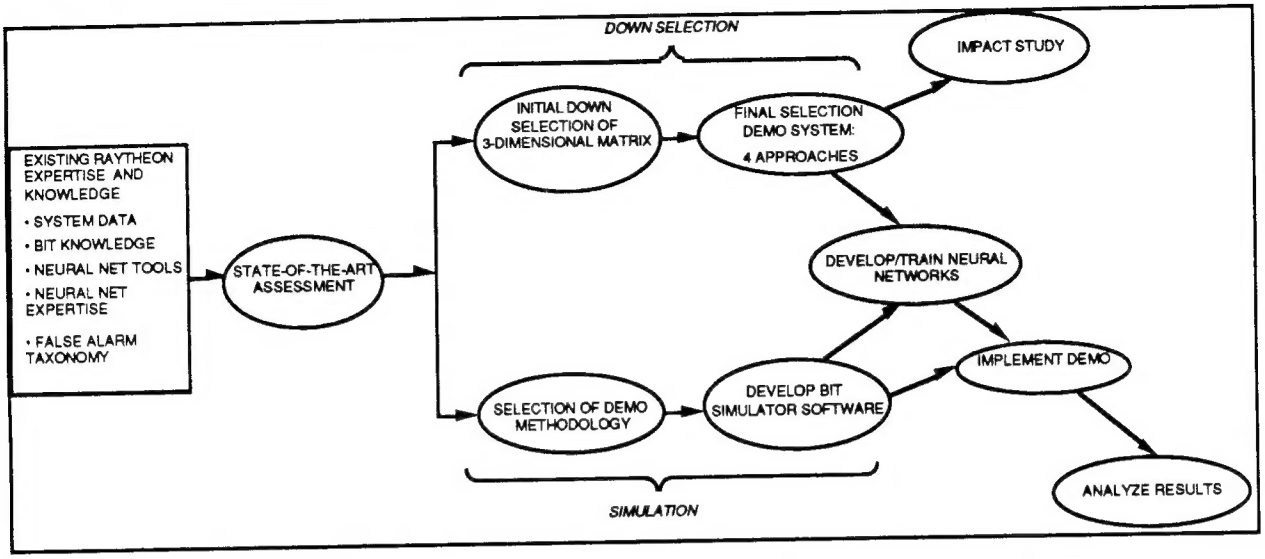
The primary goals of the NNFAF contract were to:

1. Improve the performance of BIT;
2. Filter false alarms and identify intermittent failures;
3. Utilize maturing neural network technology; and
4. Assess the impact of neural network technology insertion in fielded avionics systems.

A flow diagram of the overall NNFAF investigation methodology is shown in the figure below.

The NNFAF effort consisted of the following major tasks: State-of-the-Art (SOA) assessment, demonstration approach down selection, data generation using a custom-built BIT simulator, neural network development, performance of experiments, analysis of results, demonstration of results, and impact study. Current literature in the areas of BIT, false alarms, and neural networks were reviewed during the SOA assessment. The SOA assessment resulted in lists of all candidate BIT techniques, neural networks, and fault report causes (FRC) that could potentially be used in a detailed analysis. The down selection task reduced all possible combinations of neural networks, BIT, and fault report causes to the four most desirable demonstration approaches. The down selection task consisted of three phases using practicality, feasibility, commonality, and general

weighting criteria to choose candidate combinations of BIT techniques, fault report causes, and neural networks.



Overview of NNFAF Project and Investigation Methodology

A simulator tool was designed to model the selected BIT techniques. It produced BIT fault reports in response to fault report cause inputs (such as excess temperature). The BIT simulator was based on a typical system design, employing the types of BIT under investigation. The BIT fault reports were used to develop, train, and evaluate the selected neural networks. Finally, a Life Cycle Cost (LCC) impact study was conducted to assess the cost/benefit tradeoffs of neural network technology insertion at the platform, equipment, and logistics system levels.

The deliverables of the contract were:

- Monthly R&D Status Reports
- Software Design Document (2167A)
- Software User's Manual (2167A)
- Quarterly Contract Funds Status Reports
- R&D Test and Acceptance Plan (Demo Plan)
- Software
- Final Report

Report Organization and Content

This report is divided into two volumes. Volume I contains the main body of the report, and documents the conduct of the NNFAF project. It is organized to follow the sequence of tasks which were performed on the contract. It is divided into major sections which describe each of the major tasks. Appendices are used to contain exemplary or backup data which is too large or too detailed to be included in the main body of the report. Volume II contains the largest of the appendices. It may be omitted when reading the report without detracting from its overall comprehension.

The following table summarizes the report content.

Volume Number	Section Name	Description
Volume I	1. Introduction	Provides background, objective of effort, report content
	2. State-of-the-Art Assessment	Describes the state-of-the-art assessment task for BIT and neural network technologies
	3. Down Selection Process	Describes the structured method of selecting demonstration approaches via three down selections
	4. Target System	Describes the basis for fault modeling and BIT simulation
	5. Development Methodology	Describes the software development methodology and the neural network development methodology, including software tools, neural network models, neural network input data generation, and neural network training and testing
	6. Neural Network Results	Presents the results for each of the four neural networks and analyzes network performance
	7. Impact Study	Describes the conduct and results of the impact study
	8. Demonstration	Describes the plans for and the conduct of the acceptance test and software demonstration
	9. Conclusions	Presents conclusions regarding the overall outcome of the work
	10. Lessons Learned	Identifies major lessons learned on the project
	11. Recommendations for Future Work	Proposes ideas for future work in the general problem domain
	Appendices	BIT and neural network bibliographies, literature abstracts, and definitions of terms, examples of simulated BIT fault signatures, simulated fault report causes, and neural network input data files,
Volume II	Appendices	BIT and neural network tutorials on the BIT techniques, fault report causes, and neural network models investigated in this effort

1. INTRODUCTION

1.1 Background

Research in the domains of neural networks and Built-in Test (BIT) has highlighted a need to examine the potential of the neural network technology to the solution of long-standing BIT problems such as false alarms and intermittent failures. Characteristics of the BIT problem domain include time-sensitive data, temporal reasoning, pattern matching, the need to adapt to changes in the data input, and the importance of remembering history (the conditions under which a failure or non-failure occurred in the past). Certain neural network models also exhibit similar characteristics. Neural network technology has reached a sufficient level of maturity and stability such that it can be studied for its applicability to real world problems. The NNFAF project was contracted to investigate that applicability with respect to its potential for improving BIT performance.

1.2 Objective

Rome Laboratory has sponsored research into the development of AI-based techniques for improving the performance of BIT. The Smart BIT program used neural network technology as one of its approaches. Other neural network technologies needed to be investigated to identify additional promising approaches for BIT false alarm filtering. The objective of this effort was to identify, refine, and develop such techniques and demonstrate their concepts. In addition, an objective was to make recommendations for the implementation of these techniques into fielded systems. The approaches were investigated for application to various BIT methods, including those associated with the test, diagnosis and condition monitoring of a MILSATCOM system.

1.3 Approach

A vast amount of information needed to be applied to the neural network false alarm filtering project including: system knowledge, logistic knowledge, knowledge of BIT techniques, false alarm failure methods, knowledge about neural network (NN) architectures, characteristics and tools. This knowledge was an input to a state-of-the-art assessment of NN and BIT. The assessment generated a three-dimensional matrix of possible approaches to applying NN to BIT while considering fault report causes. This matrix or cube is shown in Figure 1.3-1.

A methodology based on successive down selections guaranteed that maximum effort was expended on the most promising approaches, but that all approaches received some attention. Parallel with the first down selection, the target system and methodology were finalized. The recommended target system was a MILSTAR like system with a hierarchy of NN/BIT structures. After selection of the target system, the final down selections of candidate approaches were made, eliminating any methods that could not be effectively demonstrated using the target system.

The demonstration methodology was selected, as well as the platform/tool combination of the demonstration, and the "look and feel" of the machine-interface. Demonstration criteria and expected results were established. Drafts of the Software Design Document, the R&D Test and Acceptance Plan, and the Software User's Manual were generated at this time. The implications of the selected approaches to the target system were assessed, including integrated logistics support, life-cycle costs and host hardware and software.

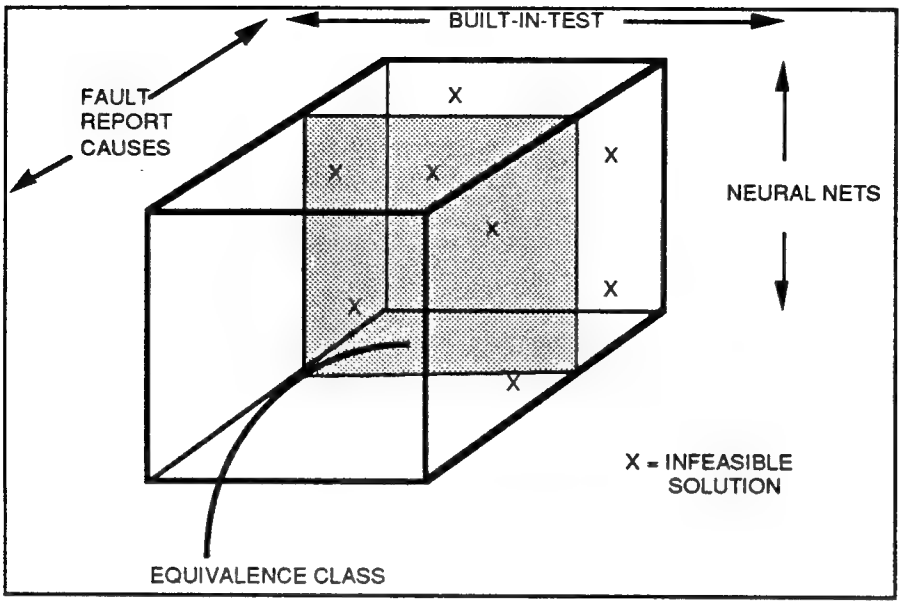


Figure 1.3-1. Application Approaches

The results of the down selections, coupled with the selection of the target system, allowed us to design, code and test the demonstration system. The appropriate NNs were trained with system data augmented to include environmental data. The system was demonstrated at Rome Laboratory, with a presentation of the expected paybacks of applying NN to BIT. Our efforts were documented in a final report.

Figure 1.3-2 graphically depicts the selection process. Subsequent sections provide more detail to many of the processes in the diagram.

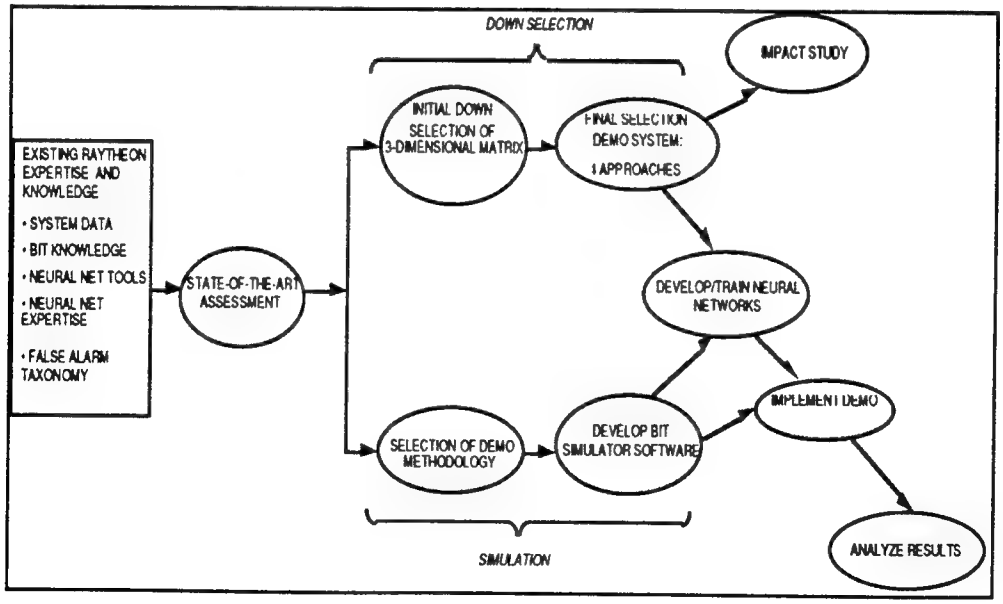


Figure 1.3-2. Overview of Approach Method

1.4 Report Organization

This report consists of an Executive Summary and Introduction, followed by descriptions of each of the major phases of the project and information relevant to each phase. The major phases are:

- State-of-the-Art Assessment
- Down Selection of Final Approaches
- Identification of Target System and BIT Simulation Mechanism
- Neural Network / Software Development
- Results Analysis / Impact Study
- Final Demonstration

2. STATE-OF-THE-ART ASSESSMENT

2.1 Built-in-Test

2.1.1 Overview/Conclusions

A study of the state-of-the-art in the area of Built-in Test (BIT) was performed. Literature was reviewed that contained material on new BIT techniques, BIT false alarm filtering techniques, BIT false alarm determination, and new BIT maintenance philosophies. The literature was selected by performing a literature search of all current journals, articles, reports, and books in recent years which contained key words. Abstracts of the literature matching the key words (such as false alarm) were gathered, and copies of the relevant articles were collected and reviewed by the NNFAF researchers. All of the articles reviewed in the BIT state-of-the-art literature search are included in the BIT Bibliography and Literature Abstracts in Appendix A.

Many interesting methods have been proposed to analyze false alarms based on mathematical or probabilistic models where found during the literature search. The crucial items in such models are the definitions of a false alarm, intermittent failure, hard failure, and a functional system. The distinctions between these terms affect the design of any neural network filtering techniques.

We have defined these terms as follows:

1. False Alarm: A False Alarm is an incorrect report to the operator that a maintenance or repair action is necessary when no action should be taken.

2. Intermittent Failure: An Intermittent failure is a fault which results in a failure that occurs repeatedly over time. Intermittent failures can occur at regular intervals or randomly.

3. Hard failure: A hard failure is the inability of an item to perform its specified function, which results in the inability of the system to perform its specified function.

4. Functional system: A Functional System is a system which performs to its specified requirements.

Other standard BIT-related terms and definitions are listed in Appendix B.

Literature describing the relationship between false alarm causes and their impact on BIT routines and reports was not found. Previous research in the areas of false alarms and filtering techniques

has developed both simplistic and sophisticated models of false alarms causes. However, these models were developed based on assumption and "experts' opinions". They were never validated (it may not be possible to do so). The problem of modeling false alarm signatures was recognized during the NNFAF project. Even if accurate false alarm signatures did exist for one system, these signatures might be dependent on the specific system or operational environment. Therefore, it was agreed that false alarms can manifest themselves in many possible signatures, depending on the system and the application. As a result, the NNFAF focus on false alarm and intermittent failure signatures is on distinguishing one from the other and not on signature definition.

The state-of-the-art assessment was considered an evolving process which continued throughout the entire contract period of performance. As new entries were encountered, the bibliography and abstracts were updated, so that the final BIT Bibliography and BIT Literature Abstracts in Appendices A and B represent the current state-of-the-art of BIT technology.

2.2 Neural Networks

2.2.1 Overview/Conclusions

A study of the state-of-the-art in the neural networks domain was performed. The purpose of this state-of-the-art assessment was to "identify, define and rank those (neural network) techniques which have application potential for improving the performance of BIT." Literature was reviewed that contained material on neural networks research areas, theory, applications, architectures, and modeling techniques. The literature was selected by performing a literature search of current journals, articles, reports, and books, as well as by contacting prominent neural network researchers directly. Keywords were used to direct the literature search. Copies of the relevant literature were obtained, read, reviewed, and annotated. The result of this effort was an extensive neural network bibliography (initially 41 entries), accompanied by abstracts which summarized each entry and noted its relevance to the contract effort.

The neural network review and assessment confirmed our neural network knowledge. A set of 22 neural network models was compiled, which represented the current state-of-the-art in the domain. No new network models were discovered. Also as part of the assessment, a neural network characterization matrix was developed, which presented each of the neural network models, sorted by major features or characteristics. This characterization matrix was prepared for use during the down selection process (selecting candidate approaches for demonstration). In addition, a collection of relevant definitions of terms was compiled, including brief descriptions of each of the network models.

The state-of-the-art assessment was considered an evolving process which continued throughout the entire contract period of performance. As new entries were encountered, the bibliography and abstracts were updated, so that the final Neural Network Bibliography and Literature Abstracts in Appendix C represent the current state-of-the-art of neural network technology.

Appendix D contains neural network definitions which were compiled during the neural network state-of-the-art assessment.

3. DOWN SELECTION PROCESS

The first down selection used practicality as a down-selection criteria. The initial Cartesian cube had almost 40,000 entries. In analyzing these approaches, we eliminated all methods that were grossly infeasible. In addition, we sought equivalent classes within the cube, such that selecting

any member in the equivalent class would adequately demonstrate the benefits of the approach within the class. Consequently only one member of an equivalent class would be selected. Using these rules we selected slightly over 500 approaches to consider further.

Results of the first down selection were reviewed with the customer. A rich space of approaches was still available and a second down selection was required. In the second down selection, an initial qualitative screening was made for system characteristics. We qualitatively rank ordered the remaining approaches and noticed that some of the highest ranked approaches were very similar. We selected the top 30 "different" approaches that demonstrated significant payback. A third down selection used quantitative characteristics and target-specific information. Any approach that could not demonstrate system payback was eliminated.

3.1 Down Selection 1

This section describes the process for the first demonstration candidate down selection.

3.1.1 Initial Matrix

At the start of the first down selection, thorough lists of BIT techniques, fault report causes, and neural network models were compiled, to identify the members in each dimension of the Cartesian cube of possible demonstration candidates.

3.1.1.1 BIT

A list of BIT techniques was generated. This list includes all current types of BIT techniques. Each technique is defined below.

Parity Check. An extra bit (parity bit) is added to data being stored or transferred. This bit is set to a value that, when added to the data bits, will always add to an odd or even number for odd or even parity respectively. When data is received of read that contains parity, all bits are added and the sum is used to verify correct parity.

Extended Parity Check. This technique is the same as parity check above, but uses multiple parity bits. It is more accurate than using only one bit of parity.

Longitudinal/Vertical Redundancy Check (LRC/VRC). An LRC/VRC check is similar to performing parity on a group of data being transferred or stored. An LRC bit is the same as a simple parity check. An LRC bit is added to each data word. A VRC word is added to the group of data words. Each bit in the VRC word serves as parity for that bit location for the group of words. Therefore, when bit 1 of the VRC is added to bit 1 of each word, the sum will be even or odd, depending on the defined VRC parity. The error rate is reduced 2 to 4 orders of magnitude when both LRC and VRC are used together.

Cyclic Redundancy Check (CRC). A CRC is usually performed on stored data but can be used on data being transferred. A word is generated and added to a group of words. When the group of words is divided by this additional word there is no remainder. This is a very accurate test.

Watchdog Timer. A device monitors a function, usually a processor. It checks for activity within a defined period of time. If no activity exists within the specified time, then a failure signal is generated. This signal can be monitored by another processor and reset to check for activity again.

Activity Detector. A device monitors one or several signals. It is monitored and reset externally. Once the device (activity detector) is reset, then any high to low or low to high transition (defined by implementation) will set the device output to an active state. This state is monitored by a processing function and reset. The active state verifies activity.

Detect Coolant Pressure Loss. This represents a mechanical switch that is triggered when a failure state exists. The signal is monitored by a processing function. Once the failure state is inactive (regained pressure), then the signal will report a functional state. This technique is used to represent any similar type of technique that uses mechanical switches.

Signal Monitor. A characteristic of a signal is checked and verified within a defined time period. This is similar to an activity detector, but the characteristic being verified can be more elaborate than a transition. Examples of what may be verified are a certain number of transitions or the length of time that the signal is in a high state.

Checksum, Logical Or. An additional word is added to a group of words, usually in a memory bank. When this word is logically ored to every data word in the memory device or bank then the sum will equal all logical ones or zeros.

Checksum, Arithmetic Addition. An additional word is added to a group of words, usually in a memory bank. The additional word is set equal to the arithmetic sum of all data words in the memory device or bank. An addition is performed on the memory bank and the results are verified against the checksum word. This test can use just one checksum word for the sum. However, it is more accurate if two words are used to maintain carry bit history during the addition.

Walking Ones. This is a test performed on a Random Access Memory (RAM) device or bank. A word is written to each memory location starting with all zeros and a one in the least significant bit for the first memory location. The one is moved over to the next higher bit and replaced by a zero for each subsequent location. The result appear as all zeros being written to every location with a one moving across the group of locations.

Six Pass Modulo 3. Six write patterns of three data words are written and read to a RAM device or bank. The patterns that are written to the RAM devices are as follows:

PASS 1	100100100100...	Address 0
	010010010010...	Address 1
	001001001001...	Address 2
	100100100100...	Address 3

(address 4 to the last address continue this sequence of 3 patterns)

PASS 2	010010010010...	Address 0
	001001001001...	Address 1
	100100100100...	Address 2
	010010010010...	Address 3
	...	...
PASS 3	001001001001...	Address 0
	100100100100...	Address 1
	010010010010...	Address 2
	001001001001...	Address 3
	...	...

PASS 4 - 6 These passes are the inverse of PASS 1 - 3.

Small Block Write/Read Tests. This test is performed on portions of RAM devices. Alternating zeros and ones (hexadecimal A) and alternating ones and zeros (hexadecimal 5) are written to a block of RAM device locations and verified. This is repeated on each pass for a new block in the RAM.

Boundary Scan. Boundary scan devices are designed with a special port that allows every functional pin to be placed in a boundary scan test mode. Every I/O of each boundary scan device can be controlled and monitored using the boundary scan port. The boundary scan port is serial with boundary scan devices linked in a large serial chain. Boundary scan tests are performed by serially shifting instructions and data to boundary scan devices on the boundary scan port. The instructions are executed and the results are serially shifted out the boundary scan port and verified.

Built-in Logic Block Observer (BILBO). A BILBO device or circuit is placed at the inputs and one at the outputs of the circuit under test (CUT). These devices pass input and output data to and from the CUT during normal operation. Each BILBO device can be externally controlled to initialize a seed at the input and output BILBO devices. The input device can function as a pseudo-random number generator and replace the normal inputs with random data. The output BILBO device can operate as a signature analyzer. After a predefined time the signature from the output BILBO device can be shifted out for verification against a known good signature.

Crosscheck. This is a technique that enables logic devices within a VLSI component to be monitored by crosscheck circuitry within the VLSI. When used with boundary scan (to access data), crosscheck can be used to sample data within a VLSI. This data can be verified against known good data.

Set Scan Design. Registers within a device act as normal registers during functional operation. However, they are configured as a serial shift register with external control in test mode. Therefore, data can be serially shifted into all registers within a device, applied to the device as in normal operation and sampled at the registers, and then serially shifted out for comparison against known good data.

Set Scan Shadow Register. Very similar to Set Scan design; however, the shift registers do not have to be normal operational registers. These registers can be external to the functional circuit, with the ability to sample data within the circuit. The test would be the same, except that these registers can be used to sample data without affecting normal operation.

Test Channel. Known test data is sent down an operational port or processing path during unused operational time. In other words, test data is time multiplexed with normal operational data.

Processor Functional BIT Routines. Functional BIT routines that are controlled by a processor are periodically performed during normal processing.

Loopback. Known test data is sent out a data path or port and is looped back on the same port or another path or port for verification. This routine is periodically performed when the data path or port is not in use.

Signature Analysis. A routine is initiated that causes a sequence of data to be generated by the circuit under test. This data is sampled and is compressed to generate a signature. The signature is

verified against a known good signature after the data sequence is complete. There are several methods of compressing the data. The most popular is to use a Linear Feedback Shift Register (LFSR).

Off-Line BIT. BIT routines are perform one at a time while the system is not in operation. Off-line BIT is performed during power up and as a special routine.

Residue Code. Data being stored or transmitted is reformatted in a code. When the data is read or received, it is decoded. If an error occurred then the decoding will detect the failure.

Viterbi Code. Similar to residue coding; however, the Viterbi code has the ability to fix erroneous bits. The coding sequence is sophisticated and is dependent on current data words read or transmitted as well as previous data.

Syndrome Code. A coding scheme similar to residue coding.

Transition Count. Monitor circuits sample signals and count the number of transitions. These circuits are sampled at a regular period and reset. If the number of transitions goes below a threshold, then an error is reported.

Analog Voltage Measurement. An analog voltage is measured and compared to a known threshold. If the measurement exceeds the threshold, then an error is reported.

Digital Signal Comparison. A digital word representing some magnitude is compared against a known threshold. If the threshold is exceeded, then an error is reported.

Power Level Detectors. An analog power level (usually an IF or RF signal) is measured and compared to a known threshold. If the measurement exceeds the threshold, then an error is reported.

Redundancy. A circuit is duplicated several times. Each redundant circuit receives the same inputs. A voting circuit is used to sample the redundant circuit outputs and send the output data that was produced by the most redundant circuits. If one redundant circuit generates data that doesn't agree with the others, then it will be ignored and the failure will not be introduced to the operational data path.

Ratio Detection. Two signals are monitored. The ratio of one signal to the other is calculated and compared to a known threshold. If the threshold is exceeded, then a failure is reported.

Statistical Threshold. A sample measurement of a portion of a circuit is performed. The statistical likelihood of the remainder of the circuit being within a defined threshold is calculated. If it is determined that the threshold is likely to be exceeded, then a failure is reported.

Several new BIT techniques were discovered during the state-of-the-art literature search. These techniques are HIT-Compress, HIT-Identical, and D-Latch Concurrent. All of these techniques involve circuitry that performs concurrent comparisons on functional circuitry. A brief description of each follows.

HIT-Compress: In this technique, a test generator circuit is used to produce a test signal that is compared to the normal inputs of a combinatorial circuit under test (CUT). When they are equivalent, a "HIT" signal is generated. The HIT signal is sent to a response verifier which

compresses the CUT's outputs and advances the test generator to the next signal. The signature within the response verifier is validated after a HIT occurs on the last test generator signal.

HIT-Identical: This technique is applied to identical combinatorial circuits such as VLSI modules. The input of several identical circuits is compared. When all inputs are the same then a "HIT" is generated. The HIT signal is sent to a circuit that compares the outputs of each identical circuit. A failure exists if the outputs are not equivalent.

D-Latch: Several techniques for testing D-latches were found. The most interesting technique is the addition of a parity tree to a D-latch bank. Parity is calculated for inputs to the D-latch bank. The parity result is stored for one clock cycle. Parity is determined for the D-latch outputs and compared to the input parity during the next clock cycle.

This list was sorted by the type of signature reported by the BIT technique, as well as the functional circuitry involved in the BIT technique. The BIT techniques were divided this way because only the BIT signature (report) and the effects of the false alarm cause on the signature can be used to analyze failures. The BIT technique tree is shown in Figure 3.1.1.1-1.

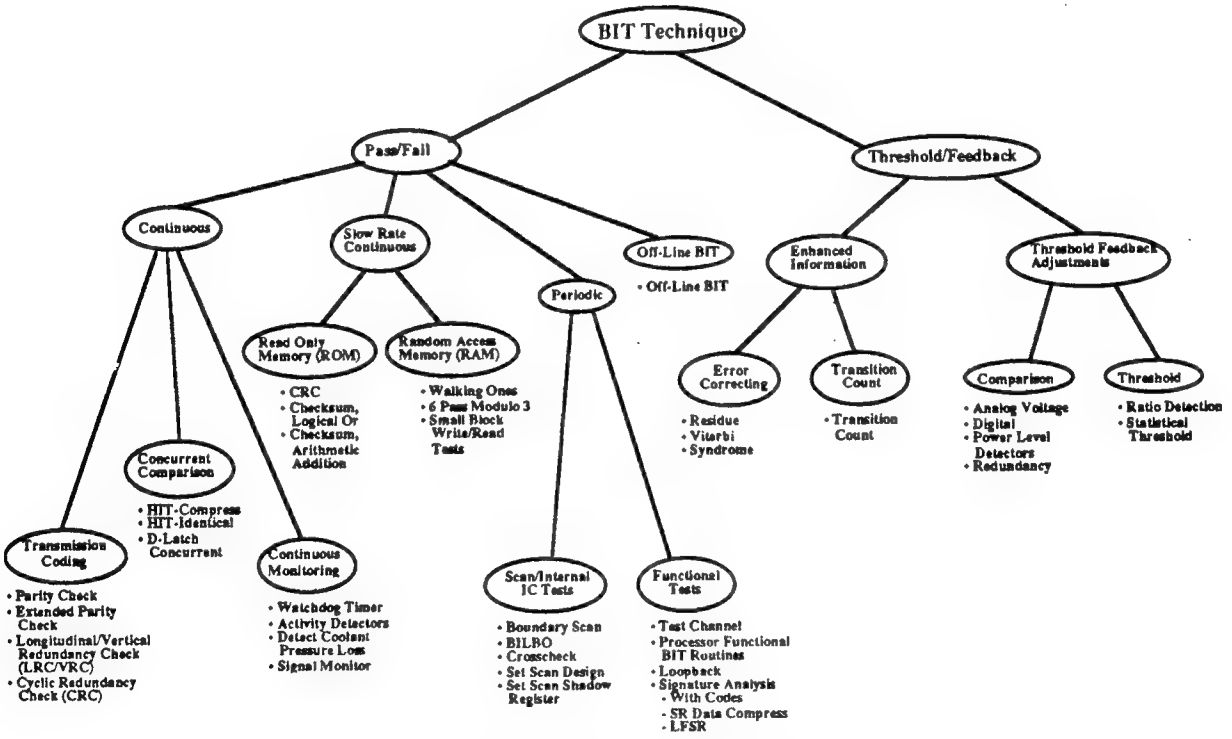


Figure 3.1.1.1-1. BIT Technique Taxonomy

The highest level of the BIT signature was broken down by the type of information reported. A BIT report is typically a pass/fail status but it can include more information, or it can be a threshold, referred to as a threshold/feedback BIT technique. The pass/fail signatures were divided by the frequency of the BIT report. The functionality of different pass/fail report BIT techniques was divided. The threshold/feedback BIT techniques were divided into enhanced information and

threshold/feedback adjustment reports. Each of the threshold/feedback divisions was further divided by functionality.

The threshold/feedback BIT techniques are new techniques that were proposed during this project. A typical BIT technique will report a pass or fail status. A threshold/feedback BIT technique report contains information that can be used by a processing function to determine the status of the BIT report. Since the processing function interprets the BIT report, it can possibly adjust the calculated BIT status based on history. It also has the potential to feed back information to the circuitry performing the BIT technique.

3.1.1.2 Fault Report Causes

A list of causes of fault reports was generated with a primary focus on false alarm causes. This list includes all items that could potentially affect the functionality of a system. They can either cause the system to appear to have a failure when it doesn't, to have an intermittent failure, or to have a hard failure. Brief definitions of fault report causes are given below.

Instantaneous One-Time Fault, Alpha Particles. This fault occurs in memory cells due to an alpha particle modifying the data within the memory cell. This false alarm is not common in ground based systems since many alpha particles do not exist within the earth's atmosphere. This situation is considered a false alarm because the hardware is fully functional; only a data bit was lost due to an outside influence.

Instantaneous One-Time Fault, Electro-Magnetic Interference (EMI) / Radio Frequency Interference (RFI). Hardware or data transmission is disturbed by EMI or RFI only for an instant of time. Note that in order for this to be a false alarm, the EMI/RFI must not be within the system's operating specification.

Shock/Vibration. The shaking of a system due to shock or vibration can cause connector contacts to temporarily disconnect. It can also cause crystal oscillators to fail. These circumstances may be due to turbulence, missile hits, or other external influences.

G Load. Extra force exerted on a system may cause connectors to temporarily become open. This situation could result from strategic maneuvers.

EMI/RFI. EMI/RFI can be caused by many external sources. Some examples are radars, jammers, and other systems within the platform, such as power transformers.

Platform Power. The power supply feeding a system can be affected by other systems using the same supply. As a result, the power may not be stable at certain instances, causing BIT routines to fail. However, the system may still be functional since the cause is external, unless it is specified that the system must function within the power fluctuation.

External Radiation. Radiation induced by an external source may disrupt signals or component operation. This radiation may be due to radar signals traveling through the system.

Temperature. Both failures and false alarms due to temperature are common. A fault report due to temperature can only be classified as a failure or false alarm if the temperature is known to be in or out of the specified operating conditions.

Component Drift. Component performance can drift or vary with age. This can cause the system to not meet its specified operation and will always be classified as a failure.

BIT Hardware Failure. BIT hardware can fail just as easily as functional hardware. Since the BIT hardware is a part of the system, any BIT hardware failure is considered a system failure.

Component Hard Failure. A component has the potential to have a hard failure, where it consistently fails to work. Component hard failures could result in intermittent as well as hard system failures depending on the application of the component.

Timing Margins. Timing margins for tests could be incorrectly defined and result in failure reports. These failures are real system failures because of a BIT design error.

BIT Thresholds. Similar to timing margin errors, BIT threshold errors are caused by incorrect values defined as thresholds during the BIT design. These failures are real system failures.

Firmware/Software Bugs. Some failures may be due to firmware/software design errors. These errors could cause BIT routines to fail when hardware is functional. Even though the functional hardware may work, these errors are real system errors since the system cannot meet its design.

Self-Interference. This occurs when part of a system interferes with another part. For example, reflections from an antenna transmitter could cause antenna pointing electronics to be disrupted by RFI.

Hardware Stress. Hardware stress due to temperature, G load, vibration/shock, power, signal tolerance, and EMI/RFI may cause components to operate out-of-specification. The result may induce system failures.

Test Tolerances. Some false alarms may be due to test tolerance failures due to variations in the system over time. The system may be fully functional and the BIT system may work correctly at first. However, the system behavior may change with time but still be functional.

False alarm causes can only be seen by their effect on a BIT technique signature (report). The fault report tree is shown in Figure 3.1.1.2-1. It has three fault report categories: false alarm causes, intermittent failure causes, and continuous hard failure causes.

All three fault report categories were used to define the difference between a BIT technique signature due to a false alarm cause, an intermittent failure cause, and hard failure cause. False alarm causes were divided into frequency of occurrence. Intermittent failures were divided into primary cause differences and then into frequency-related groups.

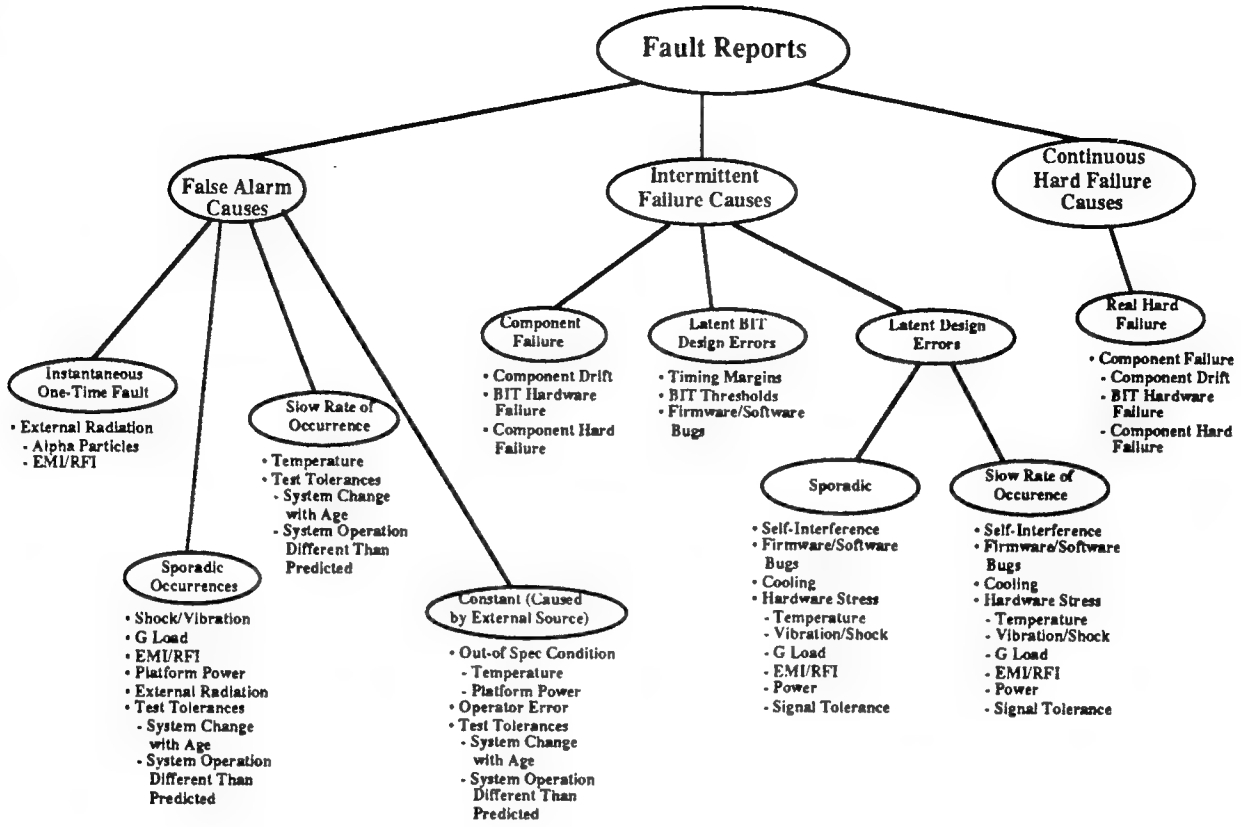


Figure 3.1.1.2-1. Fault Report Cause Taxonomy

3.1.1.3 Neural Networks

Neural network models and their characteristics are discussed in the following sections.

3.1.1.3.1 Neural Network Models

This section presents the neural network models which were selected as possible demonstration candidates based on information gained from the state-of-the-art survey. Each model is listed along with a brief descriptive summary.

1. ADALINE (ADaptive LInear NEuron, Widrow, 1959): A member of a family of trainable pattern-classifiers which distinguishes between patterns on the basis of linear discriminate functions. It can only classify linearly separable problems. The ADALINE was one of the first attempts to model biological learning.

2. ART-1 (Adaptive Resonance Theory 1, Grossbery, 1976): An unsupervised neural network model used in pattern classification. The network discovers pattern classifications on its own, in real time. It forms categories for input data, with the granularity of the categories determined by a vigilance parameter. The learning method is based on the assumption that inputs which share a greater number of features should fall into the same category. ART-1 was designed to classify binary input patterns.

3. ART-2 (Adaptive Resonance Theory 2, Grossberg and Carpenter, 1987): ART-2 is similar to ART-1 except that it was designed to classify analog inputs.

4. Backpropagation (Parker/Rumelhart, 1985/1986): A learning algorithm for updating weights in a multilayer feedforward network that minimizes mean squared mapping error (error between the designed output of the network and the actual output). It distributes the responsibility for the output error across all elements and connections by propagating it backward through the connections to the previous layer, and repeating until the input layer is reached. This is one of the most utilized network models.

5. BPTT (Backpropagation Through Time, Rumelhart, Hinton & Williams, 1986): An extension of the backpropagation learning method which can be used with problems that involve system dynamics over time. The output error is propagated back through the time path in this model. Since the propagation progresses backwards, the model requires a memory of previous time periods.

6. BAM (Bidirectional Associative Memory, Kosko, 1987): An associative memory is a memory (a network) which allows the retrieval of information by presentation of inexact or incomplete "memory keys." This network is tolerant of partial or partially erroneous (noisy) information. It is also called content-addressable. The BAM is an example of this type of network.

7. Boltzmann Machine (Ackley, Hinton & Sejnowski, 1985): A supervised learning algorithm in which network states are determined by "simulated annealing." This type of network uses a noise process to find the global minimum of a cost function.

8. Cascade Correlation (Fahlman, 1990): This is a type of supervised learning, multilayer network in which new hidden nodes are added one at a time. Its purpose is to predict the current remaining output error in the network and reduce it by creating the new hidden node. First, the new hidden node is correlated with the current network error, and then the new node is added to the network to form a cascade.

9. CCN (Compound Classifier Network, 1989): This is a network which classifies inputs against templates by a nearest neighbor algorithm. It utilizes dynamic hidden node allocation when an input is not sufficiently similar to an existing template. This model is similar to Nestor's Restricted Coulomb Energy (RCE) model.

10. Counterpropagation (Hecht-Nielsen, 1987): This is a nearest-neighbor classifier which selects from a set of exemplars (templates) by allowing them to compete against each other and selecting one winner. The winner is then decoded into a classification.

11. Hamming Network (Lippmann, 1987): This network implements a minimum error pattern classifier for binary vectors, where the error is defined using the Hamming distance. It is also called the unary model.

12. Hopfield Network (Hopfield, 1982): This is a fully-connected, feedback network which uses unsupervised learning as a pattern classifier. It can also be used to solve combinatorial optimization problems such as the Traveling Salesman Problem. It is also called the Cross-bar Associative Network.

13. Kohonen SOM (Self-Organizing Map, Kohonen, 1979-1982): This is an unsupervised model used for optimization and pattern classification. It does not require explicit training of input-output

correlations but "spontaneously self-organizes." It is used to visualize topologies and hierarchical structures of higher-dimensional input spaces.

14. LVQ (Learning Vector Quantization, Kohonen, 1988): This network assigns vectors to classes. It uses a Kohonen (SOM) layer to learn and perform the classification.

15. MADALINE (Multiple ADALINE, Widrow, 1960): This is a network of ADALINES cascaded together so that non-linearly separable problems may be addressed. It is also one of the older and potentially more restrictive models.

16. Single-Layer Perceptron (Rosenblatt, 1957): This is a trainable pattern classifier which classifies using linear discriminate functions. It is one of the original neural network models. Its goal was to model the pattern recognition capability of the visual system. It is very similar to the ADALINE. Its major weakness is that it cannot be used for non-linearly separable problems.

17. PNN (Probabilistic Neural Network, 1988): This is a neural network implementation of the Bayesian classifier statistical method. The PNN uses training data to develop distribution functions that are used to estimate the likelihood of a feature (input) being within a category (class).

18. Recirculation (Hinton & McClelland, 1988): This is an alternative to a backpropagation network in which errors are passed backwards through the feedforward connections. In this model, data is processed only in one direction, and connections are both forward and back. It uses the same learning rule as backpropagation; the connections are separated to facilitate implementation in hardware.

19. REINFORCE (Williams, 1987): This is a class of gradient-estimating algorithms for reinforcement learning. Reinforcement learning requires a reinforcement signal as training feedback (as compared to a desired output). It is an on-line learning method which can learn temporal behavior. This is one of the newer and more experimental models in the group.

20. RTRL (Real-Time Recurrent Learning, Williams & Zipser, 1989): This is a newer, more theoretical model which is recurrent and can deal with time-varying input or output.

21. Spatio-Temporal Pattern Recognition (Hecht-Nielsen, 1986): This model is a classifier used for recognizing sequences of events over time. It can be used for recognizing repetition, for example, repeating signals. It is also called the Kurogi model.

22. Temporal Difference (Sutton, 1988): A class of incremental learning procedures which are specialized for prediction (using past experience with an incompletely known system to predict its future behavior). This method assigns credit for error by means of the difference between temporally successive predictions. Learning occurs when there is a change in prediction over time.

3.1.1.3.2 Neural Network Characteristics

During the state-of-the-art assessment, a set of neural network characteristics were identified and defined, as a basis for comparison of the different models and to provide a framework for analyzing their potential contribution within the problem framework. These characteristics and their definitions are given in Tables 3.1.1.3.2-1 and 3.1.1.3.2-2. The characteristics were then applied to the neural network models, as shown in the matrix of Table 3.1.1.3.2-3. This work was done in preparation for the down selection process.

Table 3.1.1.3.2-1. Categories of Neural Network Characteristics

CATEGORY	CHARACTERISTIC
COMPLEXITY	Age, Application, Separability
CONNECTIVITY	Feedforward, Feedback (Recurrent)
LEARNING	Supervised, Unsupervised, On-Line
TEMPORALITY	With, Without
LAYERING	Single, Multiple, Hidden
STORAGE REQTS	Extremes (High, Low)
COMPUTATIONAL REQTS	Extremes (High)
TRAINING TIME REQTS	Extremes (Fast, Slow)
INPUT / OUTPUT VALUES	Binary, Any
WEIGHTS	Fixed, Adjustable

Table 3.1.1.3.2-2. Definitions of Neural Network Characteristics

CHARACTERISTIC	DESCRIPTION
AGE	The age of the neural network model could be an indication of its limitations, its degree of usefulness, or its stability. Older models may be incapable of solving complex problems; newer models may still be experimental and unproven in real world applications.
TYPE OF PROBLEM	Different neural networks are typically applied to different types of problems. The problem types fall into the following general categories: Association, Classification, Prediction, and Optimization.
FEEDFORWARD NETWORK	This is a network in which all the connections are from lower to higher layers and there are no feedback connections from one layer to another or from one layer to itself.
FEEDBACK NETWORK	This is a network in which some of the connections feed backwards through the network. Sometimes feedback is used to create time-sensitivity.
SUPERVISED LEARNING	In supervised learning, the system is trained to respond to a given input with a corresponding output by showing it the expected output.
UNSUPERVISED LEARNING	In unsupervised learning, the system receives only input stimuli. The network iteratively reorganizes itself so that each processing element responds strongly to a different set of input stimuli, forming clusters in the input space which may correspond to distinct real world concepts.
ON-LINE LEARNING	In on-line learning, the network can adapt (learn new solutions) in real-time. Other networks must be taken off-line and retrained if new (previously unseen) inputs are encountered.
TEMPORALITY	The capability of a network to deal with time-varying data, or recognize sequences of events over time.
LAYERING	This characteristic involves the hierarchical architecture of the network; networks which cannot utilize hidden or intermediate layers are typically restricted to solving linearly separable problems.
SEPARABILITY/ MAPPING	The issue of linear vs. non-linear separability is related to the issue of how a network's inputs are mapped to its outputs. A linear mapping is the most restrictive; an arbitrary mapping can represent more complex problem solutions.
LARGE STORAGE REQTS	Certain networks require extremely large memory resources, usually due either to dynamic storage allocation or a requirement for saving temporal history.
SMALL STORAGE REQTS	Certain networks require extremely small memory resources.
LARGE COMPUTATIONAL REQTS	Certain networks require extremely large computational resources, especially during the training phase.
SLOW TRAINING TIME	Slow training time means that the network must see an extremely large number of training data before it converges on a solution.
FAST TRAINING TIME	Fast training time means that the network does not require many presentations of data before it converges on a solution.
TYPE OF INPUT REQUIRED	Some networks are restricted to binary input; some may receive binary or continuous values.
TYPE OF OUTPUT REQUIRED	Some networks are restricted to binary outputs; some may output binary or continuous values.
FIXED WEIGHTS	Most networks learn by adjusting connection weights iteratively over some period of training time. However, there are a few network models in which the connection weights are not adjustable after they have been initialized. This prevents iterative learning, or generalization.

Table 3.1.1.3.2-3. Neural Network Characteristics Matrix

NETWORK MODEL	Year Intro-duced	Problem Type	Feed-fwd	Feed-back/ Recurrent	Super vised	Un-super vised	Binary Input Reqd	Tempo rality	No Hidden Layers
ADALINE	1960	Classification	X		X		X		X
ART 1	1976	Classification		X		X	X		
ART 2	1987	Classification		X		X			
Backprop	1986	Classification	X		X				
BPTT & Deriv.	1990	Classification		X	X			X	
BAM	1987	Association		X		X	X		X
Boltzmann	1985	Assoc / Opt		X	X				
Cascade Corr.	1990	Classification	X		X			X	
CCN	1989	Classification	X		X				
Counterprop.	1987	Classification		X		X			
Hamming	1987	Classification		X	X		X		
Hopfield	1982	Class / Opt		X	X		X		
Kohonen SOM	1979	Class / Opt	X			X			X
LVQ	1988	Classification	X		X				
MADALINE	1960	Classification	X		X		X		
Perceptron SLP	1957	Classification	X		X				X
PNN	1988	Class / Predict	X		X				
Recirculation	1988	Classification		X		X			
REINFORCE	1987	Theoretical	X			X		X	
RTRL	1989	Theoretical		X		X		X	
SPR	1986	Classification	X		X			X	
Temporal Diff.	1988	Theoretical		X		X		X	

Table 3.1.1.3.2-3. Neural Network Characteristics Matrix (continued)

NETWORK MODEL	Linear Map-ping	Large Storage Reqts	Small Storage Reqts	Large Comp Reqts	Slow Train Time	Fast Train Time	On-Line Learning	Binary Output Reqd	Fixed Weight
ADALINE	X							X	
ART 1				X	X		X		
ART 2				X	X		X		
Backprop			X		X				
BPTT & Deriv.		X			X		X		
BAM									
Boltzmann		X			X				
Cascade Corr.					X				
CCN		X				X	X		
Counterprop.									
Hamming									X
Hopfield									
Kohonen SOM									
LVQ									
MADALINE								X	X
Perceptron SLP	X								
PNN						X	X		
Recirculation							X		
REINFORCE							X		
RTRL				X			X		
SPR									X
Temporal Diff.							X		

3.1.2 Down Selection 1 Process

The purpose of the first down selection was to reduce the size of the solution space by one or two orders of magnitude. Since the initial solution space was so large, the first down selection was conducted iteratively. First, each dimension of the Cartesian cube was examined independently for commonality and/or infeasibility. This task is referred to as the one-dimensional down selection and was done for BIT Techniques, Fault Report Causes, and Neural Network Models. After the one-dimensional down selection, a two-dimensional down selection was conducted, to examine the BIT Technique/Fault Report Cause members for commonality and/or infeasibility. Lastly, a three-

dimensional down selection was conducted. During each iteration of the down selection, rules were formulated to describe the commonality or infeasibility which had been identified at that point in the process, and these rules were used to remove or combine members of the solution space.

3.1.3 One-Dimensional Neural Network Rules

The purpose of the one-dimensional neural network down selection process was to attempt to reduce the number of potential neural network models by reviewing them for both commonality and infeasibility, and then defining and applying commonality or infeasibility rules which would group together common models or eliminate inappropriate ones. The purpose of the commonality review was to reduce the neural network domain size by grouping models by some type of equivalence, such as an equivalent learning algorithm. The purpose of the infeasibility review was to reduce the domain size by eliminating any inappropriate models. The models were examined by using the neural network characteristics matrix (Table 3.1.1.3.2-3) which had been developed during the state-of-the-art assessment task.

The commonality review was conducted first, and it was determined that no commonality rules existed: the characteristics of the models were sufficiently unique that no equivalence could be identified at this time. The infeasibility review asked the question, "Are there any neural network models having capabilities which are technologically too restrictive, possibly obsolete?" The following characteristics were identified as exhibiting restrictiveness:

- Complexity: Linear Separability
- Layering: Single Layer Architecture
- Learning Weights: Fixed, Not Adjustable

The Linear Separability characteristic is restrictive in that a model which can only solve linearly separable problems is not flexible enough to be applied to the false alarm problem. Similarly, a network with a Single Layer Architecture which can only provide a linear mapping between input and output is also not sufficiently complex to solve this type of problem. Finally, models which do not allow adjustment of weights after they have been initialized can only learn very narrow problem solutions.

The result of the one-dimensional neural network downselection was the formulation of two infeasibility rules, based on the identification of the restrictive characteristics described above.

- **One-Dimensional Infeasibility Rule 1**: A neural network model is infeasible if it is restricted to linear mapping of input layer to output layer and does not allow indirect mapping through a hidden layer, providing no complex problem solving capability.

- **One-Dimensional Infeasibility Rule 2**: A neural network model is infeasible if its weights cannot be adjusted after they have been initialized, providing no generalized learning capability.

Upon application of these rules, the one-dimensional down selection process eliminated the following neural network models:

- **ADALINE** by rule 1 and rule 2
- **MADALINE** by rule 2
- **Single Layer Perceptron** by rule 1

3.1.4 One-Dimensional BIT Rules

The BIT techniques were analyzed for both commonality and infeasibility. A commonality rule was generated that groups BIT techniques together that will respond fault report causes with the same results or signatures. No infeasibility rules were found since each BIT technique is plausible. The commonality rule is as follows:

- **One-Dimensional BIT Commonality Rule**: Two BIT techniques are considered common if the result or report of each looks the same, and any influences that may cause the techniques to fail are the same. In other words, the BIT failure signatures must be the same. BIT failure signatures include frequency of report and the type of status (pass/fail or threshold/feedback).

This commonality rule was applied to the BIT technique tree, and resulted in the selection of 12 common groups of BIT techniques. One representative BIT technique was chosen for each group. Because of commonality, all of the original BIT techniques are represented in the 12 common groups. The result of the one-dimensional BIT technique down selection is shown in Figure 3.1.4-1.

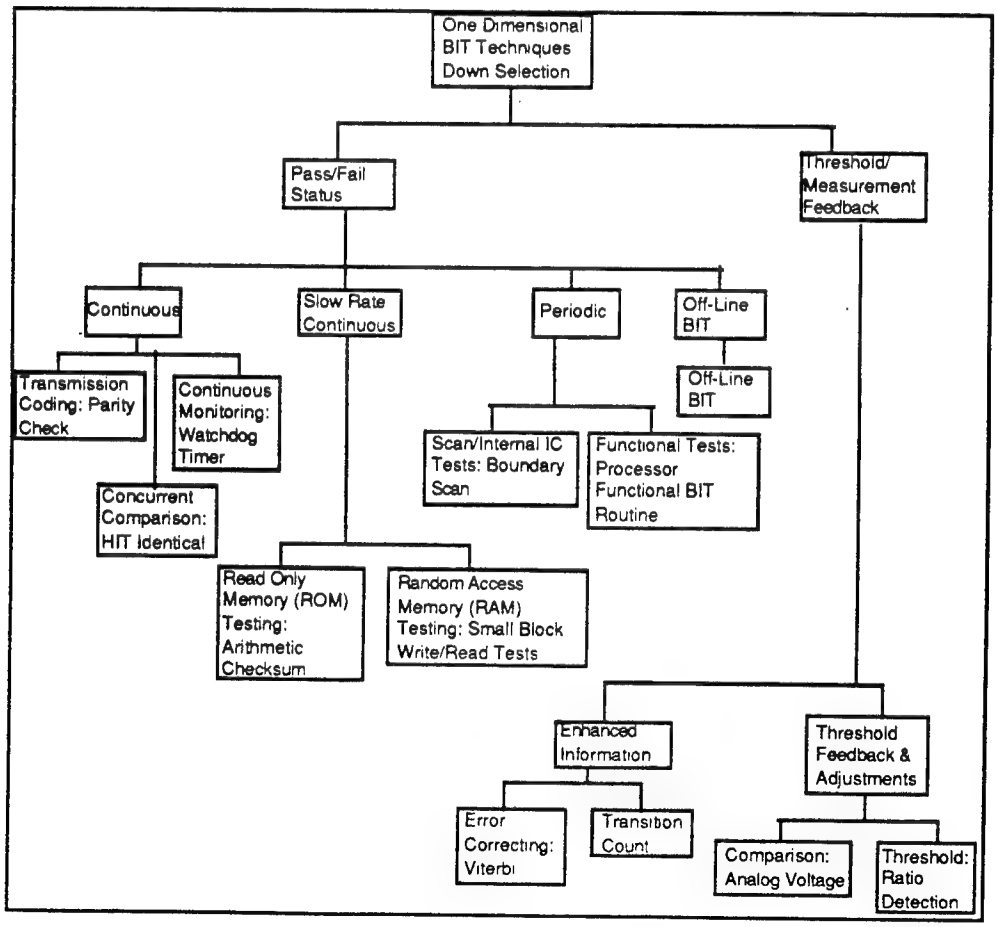


Figure 3.1.4-1. One Dimensional BIT Technique Downselection Results

3.1.5 One-Dimensional Fault Report Cause Rules

The fault report causes were analyzed for both commonality and infeasibility. A commonality rule was generated that placed fault report causes with similar signatures together. No infeasibility rules were found since each fault report cause is possible. The commonality rule is as follows:

- **One-Dimensional Fault Report Cause Commonality Rule**: Two fault report causes are considered common if a model of one cannot be distinguished from the other. The fault report models are characterized by frequency of occurrence, burstiness, collaborative signals, correlations with other parameters (system functions/signals or other BIT reports), and the effects on the system hardware.

This commonality rule was applied to the fault report cause tree, and resulted in the selection of 9 common groups of fault report causes. One representative fault report cause was chosen for each group. Because of commonality, all of the original fault report causes are represented in the 9 common groups. The result of the one-dimensional fault report cause down selection is shown in Figure 3.1.5-1.

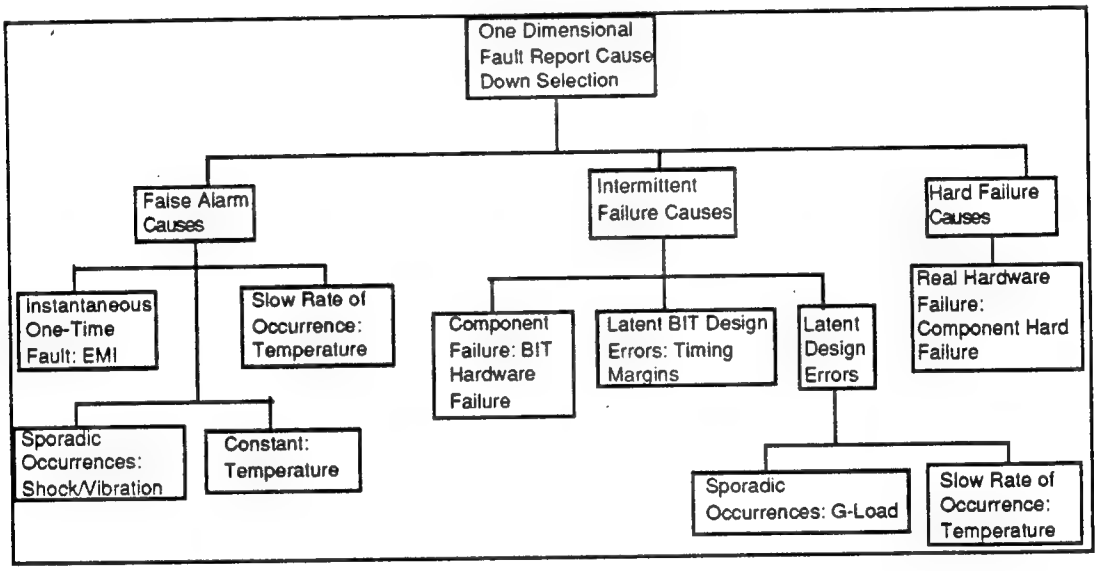


Figure 3.1.5-1. One Dimensional Fault Report Cause Down Selection Results

3.1.6 Two-Dimensional BIT x Fault Report Cause Rules

A matrix was generated using the results of the one-dimensional BIT technique and fault report causes down selection. The matrix contained 108 BIT technique vs. fault report cause entries (see Figure 3.1.6-1). Each entry was analyzed for both commonality and infeasibility. A commonality rule was generated that placed entries with similar signatures and responses to other entries together. No infeasibility rules were found, since all combinations were plausible. The commonality rule is as follows:

- **Two-Dimensional BIT-Fault Report Cause Commonality Rule**: A combination of a fault report cause and a BIT technique is considered common with a different combination of fault report cause/BIT technique if the resulting BIT failure

signature appears the same. The parameters within the BIT failure signature are frequency of report, type of status (pass/fail or threshold/feedback), and burstiness of failures. Also, if it is possible to adjust BIT parameters such as a threshold, then the manner in which adjustments are made must be the same.

This commonality rule was applied to the fault report cause x BIT technique matrix which resulted in 52 common groups. The BIT technique x fault report cause matrix with commonality groups is shown in Figure 3.1.6-1.

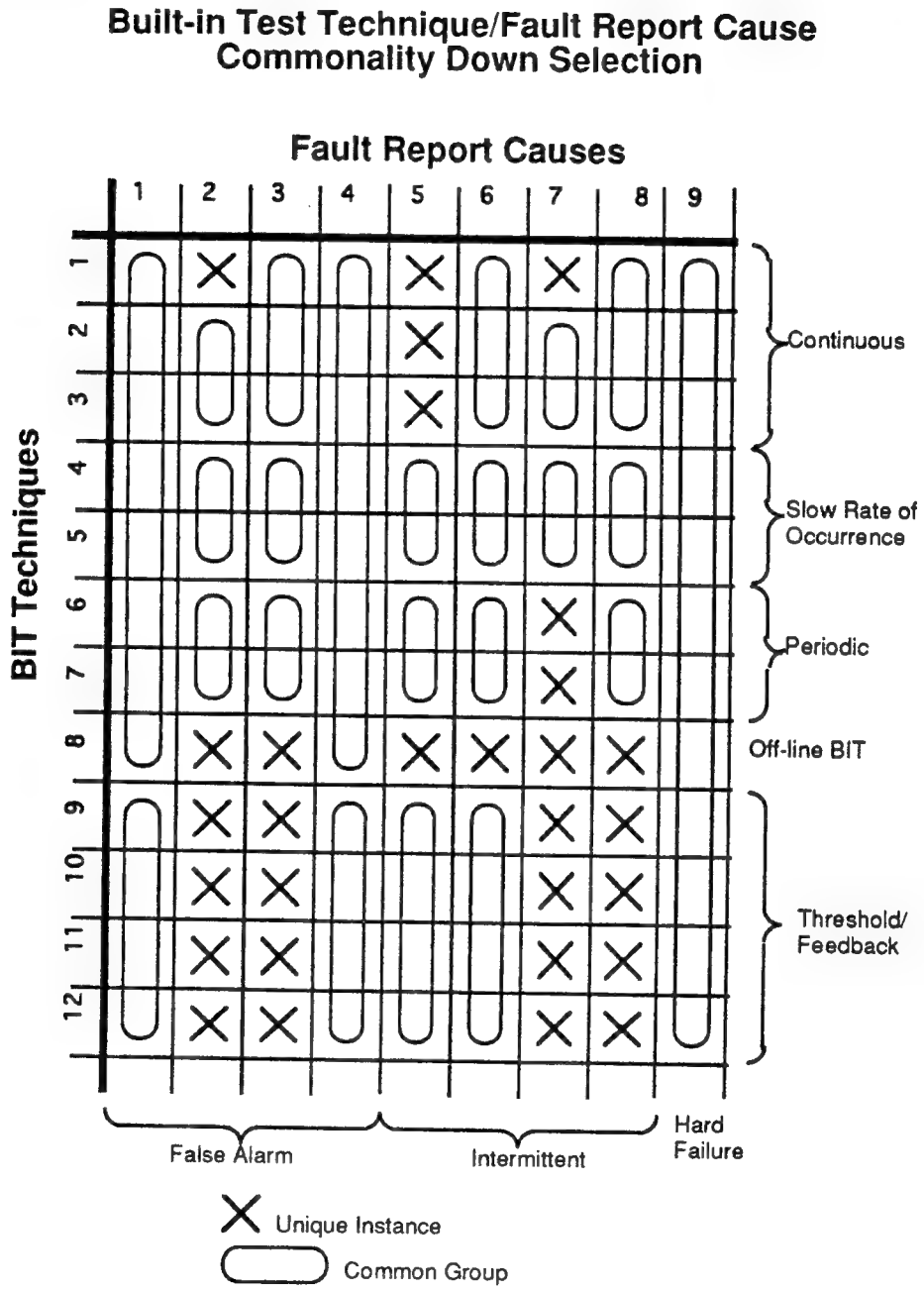


Figure 3.1.6-1. Bit Technique x Fault Report Cause Matrix

3.1.7 Three-Dimensional Rules

The purpose of the three-dimensional down selection was to integrate the results of the one- and two-dimensional selections by reviewing the characteristics of each two-dimensional Fault Report Cause / Bit Technique for infeasibility and/or commonality in the neural network domain.

The components were reviewed for commonality and it was determined that no commonality rules could be defined, due to the uniqueness of the neural network models' learning algorithms.

The components were reviewed for infeasibility. The following neural network characteristics were identified which could result in infeasibility because of their incompatibility with certain Fault Report Causes / BIT Techniques:

- a need to collect and maintain information sequences over time;
- a need for automatic learning;
- a need for adaptiveness in real time;
- a difficulty with automatic learning.

The need for collection and maintenance of information over time relates to the temporal neural network characteristic. The references to automatic learning relate to network models which use unsupervised learning; the need for real-time adaptiveness refers to on-line or real-time learning. Table 3.1.7-1 shows the neural network models as they are characterized in these categories. The three-dimensional infeasibility rules explain the relationships between these characteristics and BIT techniques/Fault Report Causes.

Table 3.1.7-1. Three-Dimensional Down Select Neural Network Model Characteristics

NETWORK MODEL	Temporal Capability	Unsupervised Learning	On-Line Learning	On-Line & Unsupervised
1. ART 1		X	X	X
2. ART 2		X	X	X
3. Backprop				
4. BPTT & Deriv.	X		X	
5. BAM		X		
6. Boltzmann				
7. Cascade Corr.	X			
8. CCN			X	
9. Counterprop.		X		
10. Hamming				
11. Hopfield				
12. Kohonen SOM		X		

TABLE 3.1.7-1. THREE-DIMENSIONAL DOWNSELECT NEURAL NETWORK MODEL CHARACTERISTICS (continued)

NETWORK MODEL	Temporal Capability	Unsupervised Learning	On-Line Learning	On-Line & Unsupervised
13. LVQ				
14. PNN			X	
15. Recirculation		X	X	X
16. REINFORCE	X	X	X	X
17. RTRL	X	X	X	X
18. SPR	X			
19. Temporal Diff.	X	X	X	X

Based on these areas of potential infeasibility, five three-dimensional infeasibility rules were defined. Each of these rules is presented below, along with its justification and an example fault scenario.

RULE 1: A neural network model is infeasible for use with instantaneous one-time fault report causes, for any BIT technique, if it does not provide temporal constructs.

RULE 1 JUSTIFICATION: An instantaneous one-time fault report cause results in one occurrence of a fault report: by definition, the fault report can only occur once within some specified time period. The neural network model must have the capability to retain and utilize fault report history over that time period, to be able to recognize that the fault report did (or did not) repeat. In this way it can verify the required single occurrence.

RULE 1 EXAMPLE: An alpha-particle induced error in one cell of a memory during a write-read RAM test.

RULE 2: A neural network model is infeasible for use with constant fault report causes, for any BIT technique, if it utilizes an unsupervised learning method.

RULE 2 JUSTIFICATION: A constant fault report cause results in a fault report which is always present. In order for a neural network to classify the report as a false alarm, external correlating data such as environmental information can be used. Typically the external data value reaches a threshold, at which point the fault report should be classified as a false alarm. An unsupervised learning model would not be capable of automatically learning the threshold point; it would need to be trained on examples.

RULE 2 EXAMPLE: Given a BIT fault signature and correlating temperature data, the fault signature will become invalid if the temperature goes out of spec. The neural network must be able to learn to classify signatures differently once the threshold has been crossed.

RULE 3: A neural network model is infeasible for use with threshold / feedback BIT techniques, for any fault report cause, if it does not utilize unsupervised and on-line learning capabilities.

RULE 3 JUSTIFICATION: A threshold / feedback BIT technique is one in which the BIT test uses a threshold value to determine the presence or absence of a fault. System operation over time can cause the threshold to degrade such that it may no longer be set to the correct value. We believe that certain neural network models can learn to recognize when a threshold is not correct and either automatically adjust it or provide appropriate feedback to the BIT system. A neural network model which cannot adaptively learn in real-time is not appropriate for this application, since the threshold behavior cannot be known in advance and is dependent upon real-time system performance.

RULE 3 EXAMPLE: A BIT detector which detects a threshold voltage and reports a fault if the voltage falls below the threshold can drift over time so that the reports become inaccurate.

RULE 4: A neural network model is infeasible for use with periodic BIT techniques, for any fault report cause, if it does not provide temporal constructs.

RULE 4 JUSTIFICATION: A periodic BIT technique requires an historical fault signature, which contains fault reports accumulated over the BIT test period. A neural network model which provides temporal constructs must be used for these situations.

RULE 4 EXAMPLE: A test channel BIT technique is segmented in time.

RULE 5: A neural network model is infeasible for use with off-line BIT techniques, for any fault report cause, if it does not provide temporal constructs.

RULE 5 JUSTIFICATION: False alarm filtering of off-line BIT techniques requires that the BIT test sequence be run more than once to simulate system behavior over time. This can be done by looping the sequence, making each individual test appear periodic, similar to an on-line periodic BIT technique. As stated in Rule 4, a periodic BIT technique requires an historical fault signature. A neural network model which provides temporal constructs must be used for these situations.

RULE 5 EXAMPLE: An off-line checksum of a PROM which is executed once at the start of an off-line BIT test sequence and is repeated only when the sequence is started again.

Rule 5 contains an important conclusion which was drawn concerning the relationship of off-line BIT techniques to false alarm filtering: false alarm filtering can only be applied to an off-line BIT technique if the off-line BIT test sequence is cycled, making it appear periodic. The results of the test sequence must be collected more than once over time in order to derive any meaningful conclusions regarding the possible occurrence of a false alarm.

3.1.8 Output Matrix

The final result of Down Selection 1 is shown in the matrix in Figure 3.1.8-1. In summary, the initial down selection process began with 37,752 candidates in the problem solution space. Through an iterative series of down selections at each dimensional level, this number was reduced to the 528 candidates which appear as darkened squares in Figure 3.1.8-1.

The following observations can be made by examining Figure 3.1.8-1. First, the initial down selection process did not completely eliminate any BIT techniques or fault report causes. The only neural network models which were eliminated were those which were determined to be technologically too primitive to be useful. In addition, the down selection process focused at this level on extremes such as instantaneous one-time fault report cause, constant fault report cause, or highly theoretical neural network models. At this first level of down selection, the researchers were careful to avoid bringing real-world constraints into the candidate selection process, and made evaluations based on inherent characteristics.

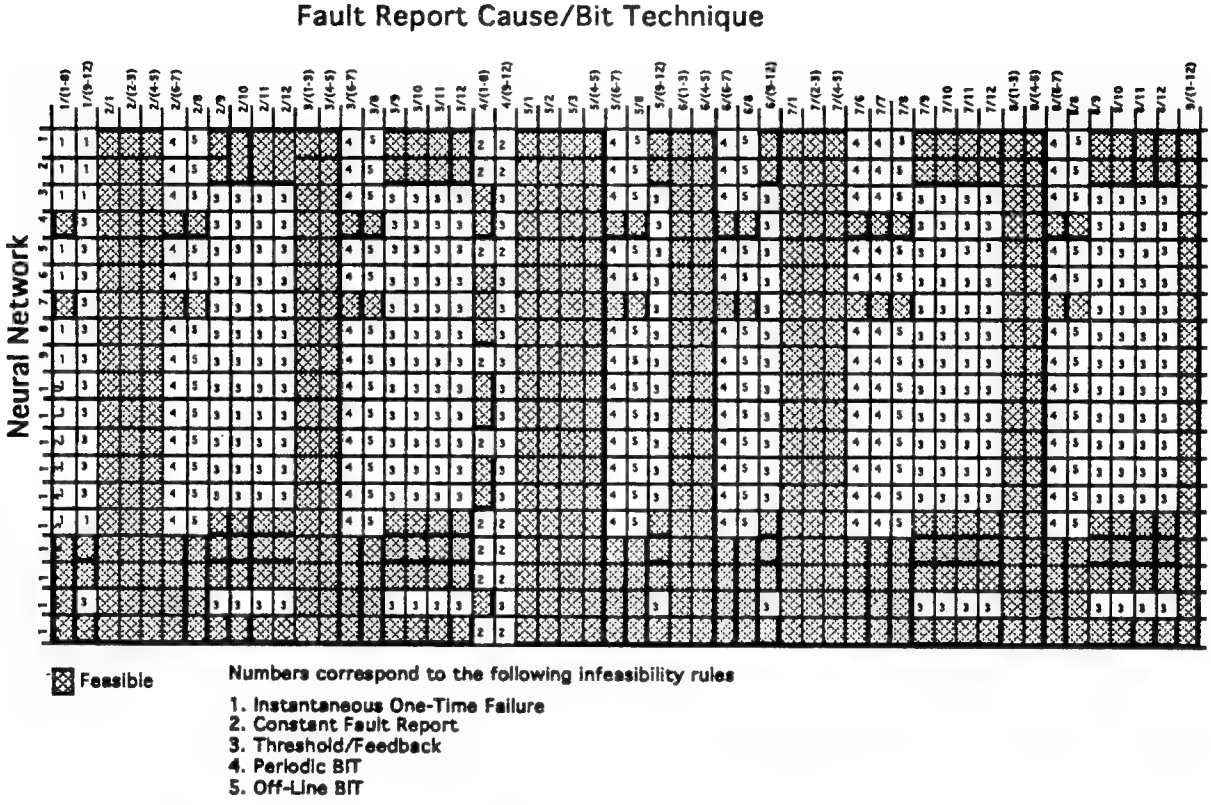


Figure 3.1.8-1. Final Result of Down Selection 1: Output Matrix

3.2 Down Selection 2

This section describes the second phase of the Neural Network False Alarm Filter Cartesian cube down selection. The first phase of the down selection reduced the Cartesian cube entries from 37,752 to 528 using infeasibility and commonality rules. The second phase of the down selection is described below. It used ranking criteria to rank order each of the 528 entries and further reduce the size of the cube.

3.2.1 List of Criteria

The first step in this phase of down selection was to generate a list of criteria by which the remaining 528 candidate solutions could be ranked. This list is shown in Table 3.2.1-1. Some of the criteria were found to be inapplicable at this stage, because they were too system-specific or

they were too difficult to judge at this point in time. The entries on the criteria list were marked with an L ("later") if they were to be deferred to the next down selection; they were marked with an N ("now") if they were immediately applicable.

Table 3.2.1-1. Second Down Selection Ranking Criteria

APPLICABILITY	CRITERION
N	Excessive Learning Time
N	Implementation Cost (hardware, software, NN, platform)
N	Environmental Feedback Required / Available
N	Configuration Management
L	Logistics Impact: Keeping Repair History
L	Logistics Impact: Effect on Lower Level Testing
L	Existing Tool Compatibility
L	Compatibility with SMART BIT / New Tools
N	Likelihood of Occurrence of Fault
N	Level of Confidence in / Reliability of NN Model
N	Memory Reqt: Long Term
N	Memory Reqt: Short Term
N	Report Latency (Real Time Constraints)
N	Batch Learning
L	Cost to Acquire Data from Simulator
N	Extent of Improvement from Present Techniques
N	Flexibility of Implementation Level
L	Extensibility to Fault Isolation (Help with FI)
L	Is the Long-term Benefit Accessible to New Systems?
N	Flexibility to Changes in System, once incorporated
L	Feasibility of Implementing NN in hardware
N	Ease of Inclusion of Feedback in Fault Signature
N	Excessive Training Data
N	Applicability of NN Model to the Problem
L	Security

3.2.2 Criteria Definitions

The 17 criteria which were applicable to this phase of the down selection were defined as follows:

1. Excessive Learning Time: There are two major focuses of this criterion. The first is that there are certain neural network models which are inherently slow learners. The second is that an adaptive, on-line learning model might take too long to become "operational" in a critical situation, such that fault reports would be invalid for an excessive length of time.

2. Implementation Cost (of entire BIT system, including the neural net enhancement): This criterion includes the amount of hardware required to implement the system, the amount of software required, the level of difficulty of neural network implementation in both hardware or software, as applicable, and platform impact. Platform impact would usually be seen in hardware or software; however, there might be some other type of impact, such as logistics. An example would be a requirement to add environmental information which would change the platform design.

3. Environmental Feedback Required: The reinforcement learning network models must have an input from the environmental, as a signal of how well the network is performing. Also, some BIT techniques will require the addition of environmental data to remove potential ambiguities in the signature classification. An example of this would be a constant false alarm, which would require some environmental feedback to distinguish it from a real fault.

4. Configuration Management: This criterion refers to the different levels of system configuration control which would result from the different neural network learning methods. A supervised learning model in which the network weights are set before the network is installed in a system, and which must be taken off-line, retrained, and reinstalled, is the most conducive to strict configuration management, since the network would be exactly the same at any site, and change control could be carefully managed. On the other hand, an adaptive, on-line learning network would evolve over time in potentially very different ways at different sites (depending upon the differences in the data presented to it), making configuration management difficult.

5. Likelihood of Occurrence of Fault Report: How common is the fault report in a real platform: a faulty report which occurs all the time is more important to correct or eliminate than one which occurs only rarely.

6. Level of Confidence in / Reliability of Neural Network Model: Certain network models can be considered more reliable than others due to their maturity and to their continued application, examination, and optimization. Others are newer, more experimental, without a significant history of success in any particular problem domain. These would have a lower level of confidence; however, they would still be considered appropriate for certain problems.

7. Long-Term Memory Requirement: This criterion addresses the situation in which an adaptive, on-line learning network which utilizes dynamic memory allocation for exemplars can gradually evolve to such a large size that the system's memory (RAM)) resources, as well as permanent storage resources, could be exhausted. The problem with this situation is that the time of occurrence cannot be predicted. Again, this is dependent upon the network model and upon the type of data it sees.

8. Short-Term Memory Requirement: This criterion refers to the level of RAM utilization in short-term system operation. It is meant to focus on both the efficiency of RAM usage of the neural

network model, as well as the risks associated with dynamic memory allocation coupled with adaptive learning, which are characteristic of some models. These models typically create new "exemplars" when an input is sufficiently different from the existing exemplars in the network. The rapidity of exemplar growth (and memory usage) is dependent upon the uniqueness of the data on which the network is learning, where more unique data will drive the memory usage up.

9. Report Latency: The amount of time it takes for the neural network to produce its output (classification). This would be a risk if the fault report were required to be seen immediately, such as a critical safety situation. This criterion could involve the architecture of the network (number of nodes, connections), as a reflection of execution time.

10. Frequency of Off-Line (Batch) Learning Required for System Updates: The requirement to take a system off-line to retrain a supervised learning neural network can be a cost risk to certain systems. This criterion addresses the situation where frequent retraining would cause adverse impact on system operation. We feel that this would be more likely to occur in situations where a representative sample of training data is difficult to collect or simulate accurately or completely.

11. Extent of Improvement from Present Techniques: The potential payback of the neural network enhanced technique.

12. Flexibility of Implementation Level: This refers to the flexibility with which the neural network can be inserted into the system. This is directly related to the level of dependence upon the frequency of fault report. At the lowest system level (BIT technique function level), fault reports occur at a higher frequency than at higher system levels where fault status is obtained by gathering fault reports from lower levels. If the classification requires a high report frequency, then the implementation would be restricted to the lower system level.

13. Flexibility to Future System Modifications: How adaptive will the neural network system be to physical changes in the system, once it has been installed.

14. Ease of Inclusion of Collaborative and/or Correlating Data in the Fault Signature: How easy is it, either in software simulation or on the real platform, to pull collaborative and/or correlating data (such as environmental data) into the fault signature, for input into the neural network. This means that the C/C data can be meaningfully embedded within the fault signature, such that it will be used by the network to learn additional information.

15. Excessive Training Data: This criterion is similar to the previous one, but focuses on the combination of a network model which exhibits slow training coupled with a requirement to provide it with an unusually large amount of data in order for it to become sufficiently trained.

16. Applicability of the Neural Network Model to the Problem: Some networks are more suited to solve certain problems than others, for example, the backpropagation network is typically used as a classifier or pattern recognizer.

3.2.3 Criteria Weighting

The criteria were weighted and categorized as shown in Table 3.2.3-1. Each of the criteria was assigned a weight, from 0 to 10, relative to the others, with zero (0) signifying lowest importance and ten (10) signifying highest importance. Each criterion was then identified as either a benefit or a cost. The sum of the benefits was made equal to the sum of the costs so that costs and benefits

would contribute equally to the overall ranking. The Report Latency criterion was found to have no impact in this down selection, so it was weighted zero, but was kept in the set for historical purposes.

Table 3.2.3-1. Second Down Selection Criteria Categorization and Weighting

CRITERION	WEIGHT	COST	BENEFIT
Likelihood of Occurrence of Fault Report	10		10
Ease of Inclusion of Collaborative / Correlating Data in Fault Signature	9		9
Implementation Cost	8	8	
Extent of Improvement from Present Techniques	8		8
Excessive Learning Time	6	6	
Environmental Feedback Required	6	6	
Short-Term Memory Requirement	5	5	
Applicability of NN Model to the Problem	5		5
Frequency of Off-Line (Batch) Learning Required for System Updates	5	5	
Configuration Management	4	4	
Long-Term Memory Requirement	3	3	
Level of Confidence in / Reliability of NN Model	3		3
Excessive Training Data	3	3	
Flexibility to Future System Modifications	3		3
Flexibility of Implementation Level	2		2
Report Latency	0	0	
TOTAL	80	40	40

Weight Scale: 1 (Lowest Importance/Impact) to 10 (Highest Importance/Impact)

Weight	10	9	8	7	6	5	4	3	2	1
# Criteria	1	1	2	0	2	3	1	4	1	0

Mean: 80 / 15 = 5.3

3.2.4 Ranking Methodology

A ranking system was used to evaluate each of the 528 candidates based on the criteria. A table was developed to aid in determining the scores for the entries. This table consisted of the neural network models (N) , BIT techniques (B), and fault report causes (FR) on one axis and the ranking criteria on the other. The N, B, and FR were evaluated separately for each criterion. An integer value from -3 to +3 was assigned to each N, B, and FR. The relative significance of the integer values is given below in Table 3.2.4-1.

Table 3.2.4-1. Second Down Selection Score Values

Value	Description
3	Exceptionally positive
2	Notably above average
1	Above average
0	Average; no significant difference from typical entry
-1	Below average
-2	Notably below average
-3	Exceptionally negative

The relationships between the N, B, and FR were considered for various N, B, and FR combinations for each ranking criteria. Equations were generated for each criterion to represent this relationship. Application of the equations to the N, B, and FR integer values resulted in an integer value from -3 to +3 with the same significance as defined above.

3.2.5 Ranking Results

A master matrix was used to rank each of the 528 candidate solutions using the ranking criteria. Each entry in the matrix received a score for each of the criteria, based on application of the equations defined in the previous step. The individual scores were summed together to provide a total score for each entry. The entries were then sorted by their individual total score to provide an overall ranking.

The 528 entries in the master matrix were initially grouped alphabetically by neural network model into 19 groups. The Adaptive Resonance Theory (ART 1 and 2) models were so similar in functionality that they were identical in criteria weighting. Therefore, the two were combined and called "ART", leaving 18 groups. When the master matrix was sorted as a whole, the resulting sorted list was very large and it was apparent that the results would be clearer if each neural network group were sorted individually, so those sorts were performed. After sorting each neural network group, the scores for all entries within each group were summed to produce 18 neural network scores. Table 3.2.5-1 shows the resulting ranking of neural networks.

Table 3.2.5-1. Neural Network Ranking Results

NEURAL NET MODEL	NN SCORE
SPR	714
BPTT	623
Backprop	596
ART	284
REINFORCE	234
Recirculation	200
LVQ	134
Cascade Correlation	69
CCN	23
Counterprop	4
Kohonen	-6
PNN	-16
Boltzmann	-72
Hamming	-74
Hopfield	-74
BAM	-168
TD	-390
RTRL	-540

Based on the neural network ranking, the top six neural network groups were chosen from the entire matrix. The quantity of six was selected because it is 50% above what is believed will be the maximum number of demonstration candidates (four). The set of six consisted of approximately 200 entries. Based on individual entry scores, the top 5 candidates were selected from each neural network group, resulting in 30 final candidates. The top 5 from each network group were selected to provide a diversity of network models. The quantity 30 was a somewhat arbitrary choice: it is a reasonable number with which to begin the final down selection, and the 30 candidates are also reasonable choices. The 30 final candidates are summarized in Table 3.2.5-2.

Table 3.2.5-2. Top 30 Summary

NN Model / Fault Report Cause / BIT Technique
ART/Slow Rate (Temperature)/Error Correcting (Viterbi)
ART/Slow Rate (Temperature)/Transition Count
ART/Slow Rate (Temperature)/Comparison (Analog Voltage)
ART/Slow Rate (Temperature)/Threshold (Ratio Detect)
ART/Slow Rate Intermittent(Temperature)/Error Correcting(Viterbi)
Backprop/Sporadic (Vibration)/Transmission Coding (Parity)

Table 3.2.5-2. Top 30 Summary (continued)

NN Model / Fault Report Cause / BIT Technique
Backprop/Sporadic (Vibration)/Continuous (Activity Detect)
Backprop/Sporadic Intermittent (G load)/Transmission Coding (Parity)
Backprop/Sporadic Intermittent (G load)/Continuous (Activity Detect)
Backprop/Sporadic (Vibration)/Slow Rate (ROM Checksum)
BPTT/Slow Rate (Temperature)/Continuous (Activity Detect)
BPTT/Slow Rate (Temperature)/Slow Rate (ROM Checksum)
BPTT/Slow Rate (Temperature)/Periodic (Boundary Scan)
BPTT/Slow Rate (Temperature)/Off-line BIT
BPTT/Slow Rate Intermittent (Temperature)/Continuous (Activity Detect)
REINFORCE/Slow Rate (Temperature)/Continuous (Activity Detect)
REINFORCE/Slow Rate (Temperature)/Slow Rate (ROM Checksum)
REINFORCE/Slow Rate (Temperature)/Periodic (Boundary Scan)
REINFORCE/Slow Rate (Temperature)/Off-line BIT
REINFORCE/Slow Rate (Temperature)/Error Correcting (Viterbi)
SPR/Slow Rate (Temperature)/Continuous (Activity Detect)
SPR/Slow Rate (Temperature)/Slow Rate (ROM Checksum)
SPR/Slow Rate (Temperature)/Periodic (Boundary Scan)
SPR/Slow Rate (Temperature)/Off-line BIT
SPR/Slow Rate Intermittent (Temperature)/Continuous (Activity Detect)
Recirculation/Slow Rate (Temperature)/Error Correcting (Viterbi)
Recirculation/Slow Rate (Temperature)/Transition Count
Recirculation/Slow Rate (Temperature)/Comparison (Analog Voltage)
Recirculation/Slow Rate (Temperature)/Threshold (Ratio Detect)
Recirculation/Slow Rate Intermittent (Temperature)/Error Correcting (Viterbi)

3.2.6 Traceability to Down Selection 1

Table 3.2.6-1 provides a traceability summary that shows which neural network models, BIT techniques, and fault report causes remain after the second down selection process. The justifications in this table serve to confirm that our selected set of 30 will provide a reasonable basis for performing the last down selection.

Table 3.2.6-1. Traceability to Down Selection 1

NEURAL NETWORK MODEL	In/Out	Comment
1. ART 1	IN	Unsupervised, on-line learning
2. ART 2	Out	Covered by ART 1
3. Backpropagation	IN	Well-known model, efficiency
4. Backpropagation Through Time	IN	Temporality with well-known parent
5. Bidirectional Associative Memory	Out	Not well-suited to this problem
6. Boltzmann	Out	Not well-suited to this problem
7. Cascade Correlation	Out	Average model
8. Compound Classifier Network	Out	Average model
9. Counterpropagation	Out	Average model
10. Hamming	Out	Not well-suited to this problem
11. Hopfield	Out	Not well-suited to this problem
12. Kohonen Self-Organizing Map	Out	Average model
13. Learning Vector Quantization	Out	Average model
14. Probabilistic Neural Network	Out	Average model
15. Recirculation	IN	Unsupervised learning
16. REINFORCE	IN	Temporality
17. Real-Time Recurrent Learning	Out	Low level of confidence in model
18. Spatio-Temporal Pattern Recog.	IN	Temporality
19. Temporal Differences	Out	Low level of confidence in model

BIT TECHNIQUE	In/Out	Comment
1. Pass/Fail Continuous Trans.Coding	IN	Flexibility of implementation
2. Pass/Fail Continuous Concurrent	Out	Commonality with 1 and 3
3. Pass/Fail Continuous Cont. Monitor	IN	Flexibility of implementation
4. Slow Rate Continuous ROM	IN	Flexibility of implementation
5. Slow Rate Continuous RAM	Out	Commonality with 4
6. Periodic Scan	IN	Flexibility of implementation
7. Periodic Functional Tests	Out	Commonality with 6
8. Off-Line BIT	IN	Considered to be like Periodic
9. Threshold Feedback Error Corr	IN	New technique
10. Threshold Feedback Trans Ct	IN	New technique
11. Threshold Feedback Comp	IN	New technique
12. Threshold Feedback Rat Detect	IN	New technique

Table 3.2.6-1. Traceability to Down Selection 1 (continued)

FAULT REPORT CAUSE	In/Out	Comment
1. Instantaneous One-Time Fault	Out	Difficult to model, not likely to occur
2. Sporadic Occurrences	IN	Likely to occur, higher payback
3. Slow Rate of Occurrence	IN	Likely to occur, higher payback
4. Constant	Out	Need collaborative data, neural net not necessary to solve problem
5. Intermittent Component Failure	Out	Difficult to model, signature would be covered by 7,8
6. Intermittent Latent BIT Design Error	Out	Difficult to model, signature would be covered by 7,8
7. Latent Design Sporadic	IN	Likely to occur, higher payback
8. Latent Design Slow Rate	IN	Likely to occur, higher payback
9. Real Hardware Failure	Out	Collaborative data helpful, neural net not necessary to solve problem

3.3 Final Down Selection

3.3.1 Final Down Selection Methodology

The purpose of the final down selection was to reduce the number of demonstration candidates from 30 to approximately 10, all of which would be equally appropriate for the demonstration, and to select the final demonstration candidates from this set. The top 30 candidates from the second down selection were examined on an individual basis, looking for quantitative reasons why they should be eliminated. In the process of looking at each candidate for advantages and disadvantages, a last set of down selection rules was formulated. These rules are discussed below.

RULE 1. The BIT feedback techniques are ideally attractive because they have the potential to significantly improve the BIT process. However, for the purposes of this demonstration, some are too complex to implement or too difficult to model with the MILSTAR simulator. The concept of these techniques would be to have the neural network learn how to adjust thresholds or test criteria within the BIT system and then feed this result back into the system, providing dynamic threshold adjustment in real time. In order to implement this concept, the BIT system software would require modification to allow for the dynamic threshold adjustment, and the MILSTAR simulator would also require modification to incorporate this functionality. The BIT techniques which fall into this category are the Comparison and Threshold techniques.

RULE 2. During this down selection, the concept of the fault signature was looked at in some detail, in order to provide a common focus for discussion. Fault signatures were proposed for all types of BIT techniques, and for the fault report causes. (Fault signatures are discussed in Section 4.1, Fault Model) As a result of this process, it became apparent that when the BIT technique is the same, certain

fault report causes due to false alarms will belong to the same family of fault signatures as fault report causes due to intermittent failures. These fault report causes can be considered in pairs since both will need to be modeled in order for the neural network to distinguish between them and classify each one correctly. These fault report pairs are Slow Rate Continuous / Slow Rate Intermittent Continuous, and Sporadic Continuous / Sporadic Intermittent Continuous. It was determined that by selecting one of these as a demo candidate, both would effectively be demonstrated.

RULE 3. As a result of examining and defining fault signatures, it was determined that the fault report signatures for Transmission Coding and Continuous BIT techniques cannot be distinguished because they have the same frequency of occurrence. Therefore, only one of these techniques should be selected, and that choice will be arbitrary, since both are effectively equivalent.

RULE 4. Continuous, Slow Rate, Periodic, and Off-Line Fault Report Causes will all have the same signature, but the frequency of this signature will be elongated in time (from Continuous to Off-Line) as the report frequency decreases. Continuous has the highest reporting frequency and therefore the best resolution of information. In addition, because the BIT system will be modeled at different levels of test (from BIT level through subsystem level to system level), it was determined that a Continuous fault report may appear to be Slow Rate or Periodic at higher system levels as the frequency of fault reporting decreases. A Continuous fault report at the BIT level may appear to be a Slow Rate or Periodic fault report at the system level, because the reports are sampled and read at longer intervals of time. Since we will implement the neural network improvements at varying reporting levels within the system, a Continuous fault report cause will also represent Slow Rate, Periodic, and Off-Line, and all may be studied and compared by modeling just one. Therefore, all but Continuous can be eliminated.

RULE 5. Two of the groups of solutions (ART and Recirculation) were identical except for neural network model. This led to a comparison of ART and Recirculation, to determine if one were more suited to the particular BIT/Fault Report Cause combination. This comparison resulted in the decision that ART should be selected and Recirculation should be eliminated. The reasons for this decision are that Recirculation has a narrower domain of applicability, typically applied to problems involving noise filtering where auto-association is required (in auto-association, the network is presented with an input and is expected to respond with the same uncorrupted input). The ART network is more complex than Recirculation, but this complexity allows it to be applied to more general classification problems, a capability which is required for our application.

Table 3.3.1-1 shows the results of this down selection. The value in the second column is the rule number (from above) which either eliminates the entry or relates it to an equivalent entry. This down selection has reduced the selection space from 30 to 8 (entries 1 and 5, 6 and 7, 8 and 9, 11 and 15, 21 and 25 are counted as one by rules 2 and 3).

Table 3.3.1-1. Results of Applying Final Down Selection Rules

Entry	Rule	NN/Fault Report Cause/BIT Technique
1	2	ART/Slow Rate (temp)/Error Correct (Viterbi)
2		ART/Slow Rate (temp)/Transition Count
3	1	ART/Slow Rate (temp)/Comparison (Analog Voltage)
4	1	ART/Slow Rate (temp)/Threshold (Ratio Detect)
5	2	ART/Slow Rt Int (temp)/Error Correct (Viterbi)
6	3	Backprop/Sporadic (vib)/Transmission Coding (Parity)
7	3	Backprop/Sporadic (vib)/Continuous (Activity Detect)
8	3	Backprop/Sporadic Int (G load)/Trans Coding (Parity)
9	3	Backprop/Sporadic Int (G load)/Continuous (Act Detect)
10	4	Backprop/Sporadic (vib)/Slow Rate (ROM Checksum)
11	2	BPTT/Slow Rate (temp)/Continuous (Activity Detect)
12	4	BPTT/Slow Rate (temp)/Slow Rate (ROM Checksum)
13	4	BPTT/Slow Rate (temp)/Periodic (Boundary Scan)
14	4	BPTT/Slow Rate (temp)/Off-line BIT
15	2	BPTT/Slow Rt Int (temp)/Continuous (Activity Detect)
16		REINFORCE/Slow Rate (temp)/Continuous (Act Detect)
17	4	REINFORCE/Slow Rate (temp)/Slow Rate (ROM Chksm)
18	4	REINFORCE/Slow Rate (temp)/Periodic (Boundary Scan)
19	4	REINFORCE/Slow Rate (temp)/Off-line BIT
20		REINFORCE/Slow Rate (temp)/Error Correct (Viterbi)
21	2	SPR/Slow Rate (temp)/Continuous (Activity Detect)
22	4	SPR/Slow Rate (temp)/Slow Rate (ROM Checksum)
23	4	SPR/Slow Rate (temp)/Periodic (Boundary Scan)
24	4	SPR/Slow Rate (temp)/Off-line BIT
25	2	SPR/Slow Rt Int (temp)/Continuous (ActivityDetect)
26	5	Recirculation/Slow Rate (temp)/Error Correct (Viterbi)
27	5	Recirculation/Slow Rate (temp)/Transition Count
28	1	Recirculation/Slow Rate (temp)/Comparison (Analog Voltage)
29	1	Recirculation/Slow Rate (temp)/Threshold (Ratio Detect)
30	5	Recirculation/Slow Rt Int (temp)/Error Correct (Viterbi)

Table 3.3.1-2 shows the recommended choices for the final demonstration. A value in the Choice column indicates that the entry is recommended for demonstration. We recommend that choice 1 be implemented to evaluate the performance of a newer, more experimental neural network model which provides a temporal capability (REINFORCE). This will also demonstrate the effectiveness of a Threshold/Feedback BIT Technique. We recommend that either choice 2A or 2B be implemented to evaluate another temporal model (BPTT or SPR). Either choice 3A or 3B could be selected since they are equivalent; either one will provide a means of evaluating the effect of Vibration on the performance of Backpropagation. Either choice 4A or 4B could be selected since they are equivalent; either one will allow an evaluation of Backpropagation affected by G-Load.

Table 3.3.1-2. Recommended Choices for Demonstration

Entry	Choice	NN/Fault Report Cause/BIT Technique
1		ART/Slow Rate (temp)/Error Correct (Viterbi)
2		ART/Slow Rate (temp)/Transition Count
3		ART/Slow Rate (temp)/Comparison (Analog Voltage)
4		ART/Slow Rate (temp)/Threshold (Ratio Detect)
5		ART/Slow Rt Int (temp)/Error Correct (Viterbi)
6	3A	Backprop/Sporadic (vib)/Transmission Coding (Parity)
7	3B	Backprop/Sporadic (vib)/Continuous (Activity Detect)
8	4A	Backprop/Sporadic Int (G load)/Trans Coding (Parity)
9	4B	Backprop/Sporadic Int (G load)/Continuous (Act Detect)
10		Backprop/Sporadic (vib)/Slow Rate (ROM Checksum)
11	2A	BPTT/Slow Rate (temp)/Continuous (Activity Detect)
12		BPTT/Slow Rate (temp)/Slow Rate (ROM Checksum)
13		BPTT/Slow Rate (temp)/Periodic (Boundary Scan)
14		BPTT/Slow Rate (temp)/Off-line BIT
15		BPTT/Slow Rt Int (temp)/Continuous (Activity Detect)
16		REINFORCE/Slow Rate (temp)/Continuous (Act Detect)
17		REINFORCE/Slow Rate (temp)/Slow Rate (ROM Chksm)
18		REINFORCE/Slow Rate (temp)/Periodic (Boundary Scan)
19		REINFORCE/Slow Rate (temp)/Off-line BIT
20	1	REINFORCE/Slow Rate (temp)/Error Correct (Viterbi)
21	2B	SPR/Slow Rate (temp)/Continuous (Activity Detect)
22		SPR/Slow Rate (temp)/Slow Rate (ROM Checksum)
23		SPR/Slow Rate (temp)/Periodic (Boundary Scan)
24		SPR/Slow Rate (temp)/Off-line BIT
25		SPR/Slow Rt Int (temp)/Continuous (ActivityDetect)
26		Recirculation/Slow Rate (temp)/Error Correct (Viterbi)
27		Recirculation/Slow Rate (temp)/Transition Count
28		Recirculation/Slow Rate (temp)/Comparison (Analog Voltage)
29		Recirculation/Slow Rate (temp)/Threshold (Ratio Detect)
30		Recirculation/Slow Rt Int (temp)/Error Correct (Viterbi)

3.3.2 Final Down Selection Results

Table 3.3.3-1 shows the four final demonstration approaches and four alternatives.

Table 3.3.3-1 Final Demonstration Approaches

Neural Network Model	Fault Report Cause	BIT Technique	Alternative Approach
REINFORCE	Slow Rate (Temperature)	Error Correction (Viterbi)	Neural Network Model: ART
Backpropagation Through Time (BPTT)	Slow Rate (Temperature)	Continuous (Activity Detection)	Neural Network Model: SPR
Backpropagation	Sporadic (Vibration)	Transmission Coding (Parity)	BIT Technique: Continuous (Activity Detection)
Backpropagation	Sporadic Intermittent (G-Load)	Transmission Coding (Parity)	BIT Technique: Continuous (Activity Detection)

By selecting these approaches, the following benefits were anticipated:

- We could evaluate the REINFORCE neural network model, which is a newer, more experimental model with a temporal capability.
- We could examine the benefits of the Threshold/Feedback BIT Technique, which is also newer and more experimental.
- We could evaluate Backpropagation Through Time as a second, more established temporal neural network model.
- We could assess backpropagation (the most widely used neural network model) in a temporal context.
- We could investigate three distinct variations of fault signatures: temperature, vibration, and G-load.

Note that the BPTT network was later replaced by the SPR network because of limitations of the commercial neural network development tool which was used on the NNFAF contract.

4. TARGET SYSTEM

4.1 Fault Model

The NNFAF contract used simulated fault data as input to the neural networks. To define the scope and functionality of the simulation process, a model was developed based on the following definitions and assumptions. The model was initially conceived at the conclusion of the final down selection task and was refined during the definition of the BIT simulator requirements.

BIT status reports are not only based on the BIT analysis and reporting circuit, but also on any circumstance that can cause a fault report. BIT circuitry usually analyzes data and reports status at a regular interval. A series of BIT device status reports over a given time interval is called a BIT fault signature (Figure 4.1-1).

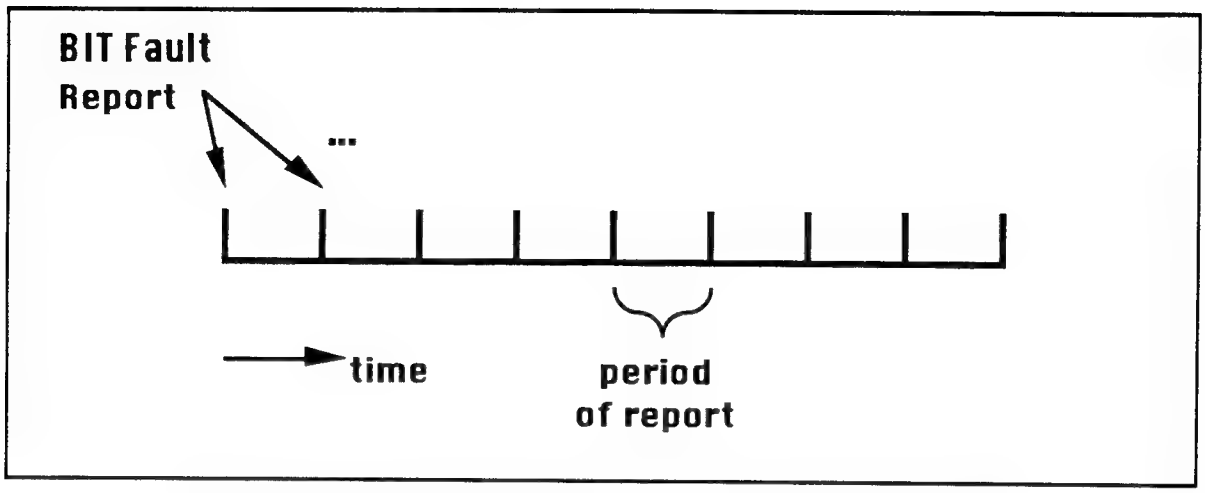


Figure 4.1-1. General Concept of BIT Fault Signature

A fault report cause is any circumstance that can make the BIT circuitry report a failure. For NNFAF, a model was implemented that contains BIT device fault signatures and a fault report cause, and represents the relationship between the two. The model was later used in formulating the requirements for the BIT simulations.

4.1.1 BIT Fault Report Signatures

During the final down selection period, we investigated BIT fault signatures and found that typical BIT fault reports have similar fault signatures which vary only in the frequency with which they are reported. Unique BIT reports, such as those from threshold/feedback techniques, do not fit this pattern. Therefore, two general types of fault report signatures were considered: pass/fail and error correcting. They are discussed below. Appendix E shows the fault signatures which were defined for the NNFAF BIT techniques.

4.1.1.1 Pass/Fail BIT Fault Report Signatures

The most common type of BIT signature is a single binary bit status report at regular intervals that represents a pass or fail state, for example, a parity BIT technique. Under normal circumstances the BIT report has a very low probability of reporting a failure. A fault report cause curve can be added to the BIT model to represent a temporal situation which changes the probability that a failure will be reported. Figure 4.1.1.1-1 represents a pass/fail BIT signature.

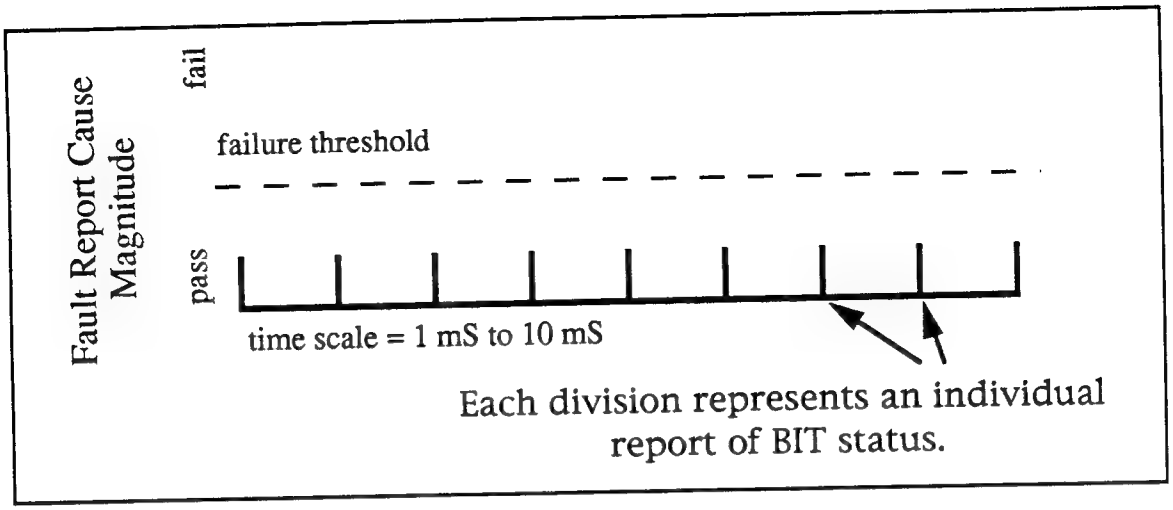


Figure 4.1.1.1-1. Pass/Fail BIT Fault Report Signature

The x axis represents time with a tick mark at each discrete BIT status report. The y axis represents the magnitude of a fault report cause. As the fault report cause magnitude increases, the probability of reporting a failure slowly increases. Eventually, a threshold is reached where the magnitude of the fault report cause begins to have a more pronounced effect on the BIT circuit and the probability of reporting a failure rapidly increases. Figure 4.1.1.1-2 is an example of what the change in probability of failure looks like as the magnitude of the fault report cause increases. Once the threshold is exceeded, the circuit performance falls off.

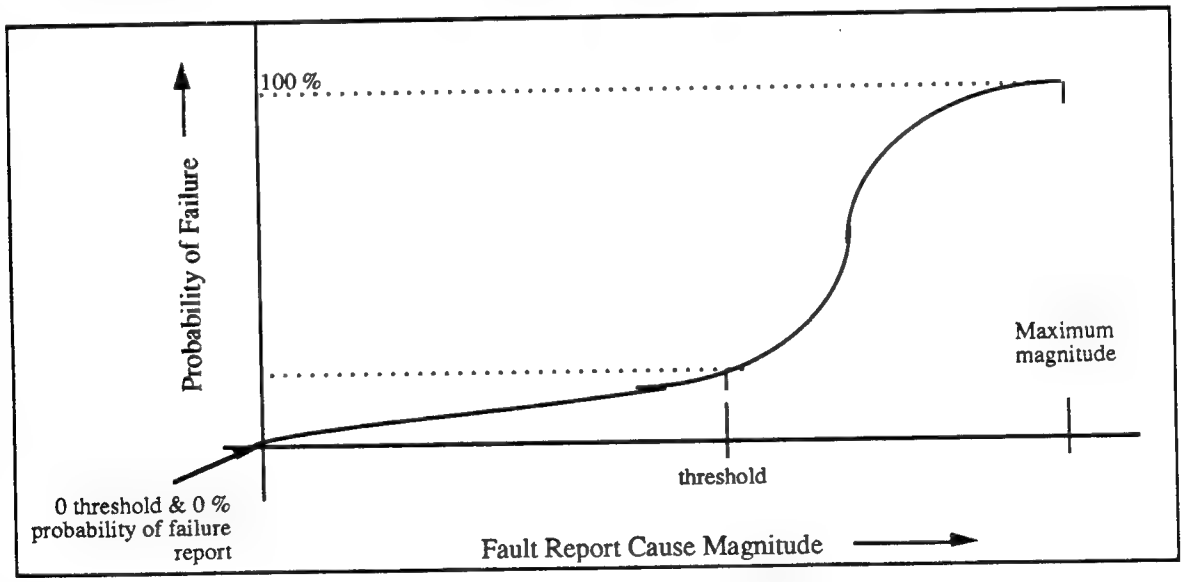


Figure 4.1.1.1-2. Theoretical Effect of Fault Report Cause on Failure Report Probability

Circuits will perform effectively while the fault report cause is beneath the failure threshold. Once the threshold is exceeded, then failures are expected.

For NNFAF, the simulator model used a simplified version of this curve as shown in Figure 4.1.1.1-3.

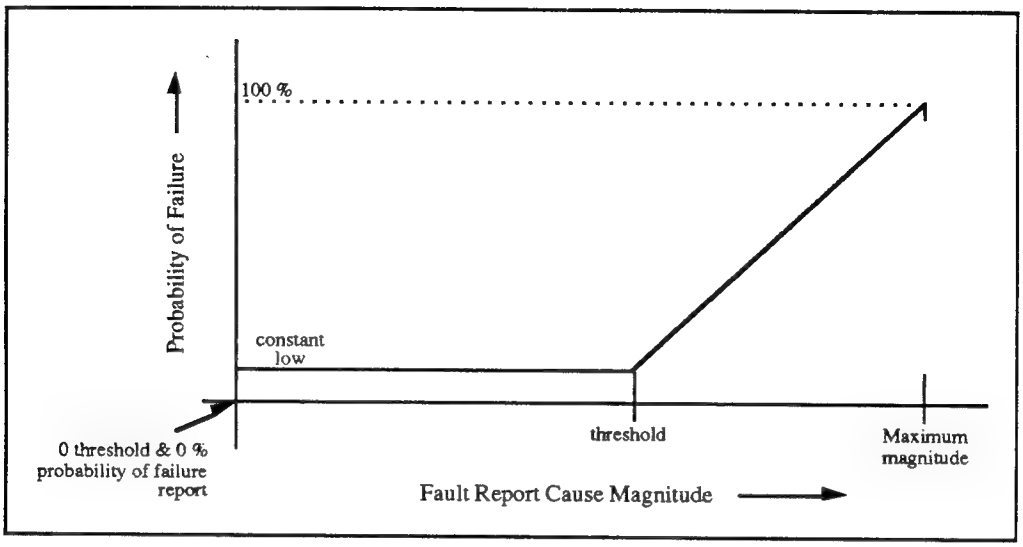


Figure 4.1.1.1-3. Effect of Fault Report Cause on Failure Report Probability, as Implemented for NNFAF

4.1.1.2 Error Correcting BIT Fault Report Signatures

An enhanced BIT report is available from circuits that perform error correction, for example, a Viterbi decoder. These BIT circuits can be designed to report three failure states:

1. Pass
2. Error detected and corrected
3. Error detected but not correctable

An error correcting BIT fault report signature is similar to the pass/fail BIT signature, except that two thresholds exist that define boundaries between the three failure states. Figure 4.1.1.2-1 shows an example of an error correcting BIT signature.

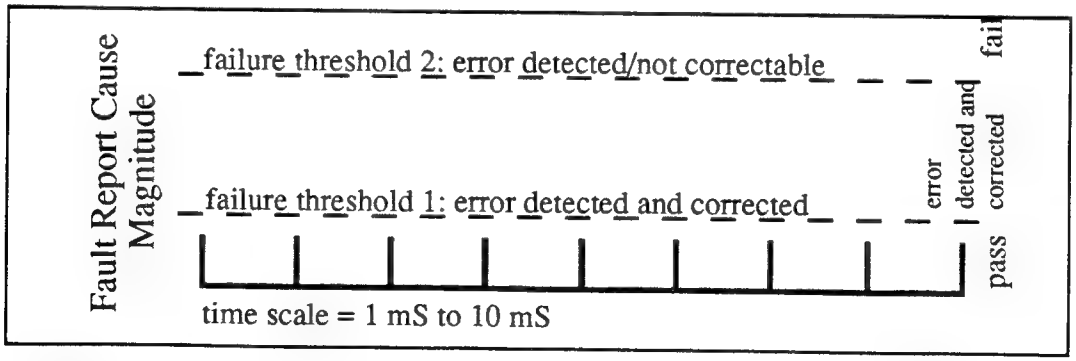


Figure 4.1.1.2-1. Error Correcting BIT Fault Report Signature

If the fault report cause magnitude is below the 'error detected and corrected' threshold, then the probability of reporting a correctable error is very low, and the probability of reporting a non-correctable error is extremely low. If the fault report cause magnitude is between the two thresholds then the probability of reporting a correctable error begins to increase rapidly, but the probability of reporting a non-correctable error is still very low. When the upper failure threshold is exceeded, the probability of reporting a correctable fault is very high and increasing, and the probability of reporting a non-correctable failure increases rapidly.

4.1.2 Fault Report Cause Model

For NNFAF, a fault report cause was modeled as a curve representing magnitude of the fault report cause over time. A threshold is defined for the curve to represent the magnitude where failure reports are expected to appear. This curve can be used to model any circumstance that will effect functional operation over time. Some examples of fault report cause curve models are temperature, vibration, and G-load. We believe that other situations can be represented as a fault report cause curve as long as their effect on a circuit can be modeled over time. The fault report cause curve definition methodology is described in Appendix F. The three fault report cause models which were selected for NNFAF (temperature, vibration, and G-load) are shown in Figure 4.1.2-1.

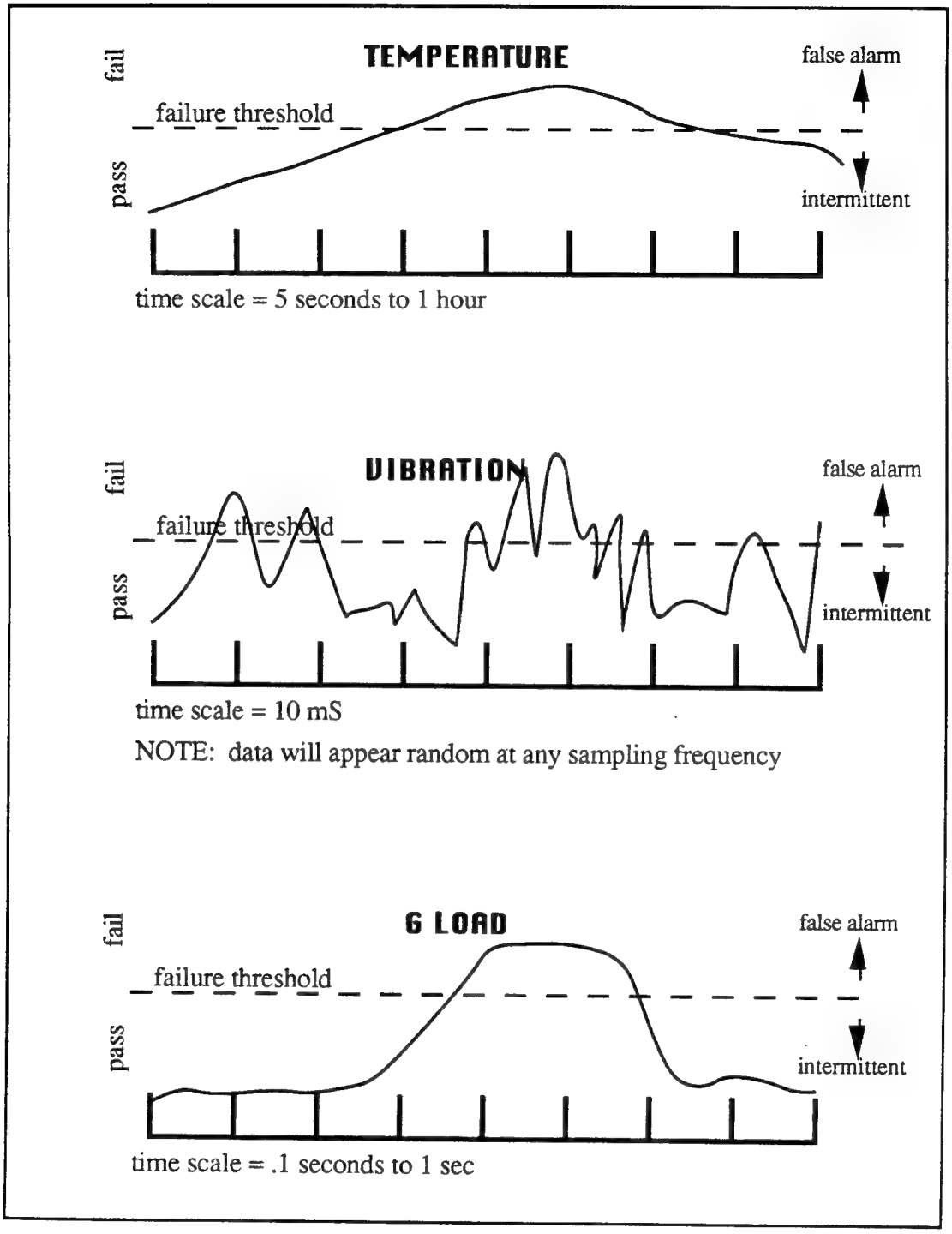


Figure 4.1.2-1. NNFAF Fault Report Causes

4.1.3 Combined BIT Signature/Fault Report Cause Model

The BIT signature and fault report cause model are combined to produce a BIT signature model based on the fault report cause. This model is produced by overlapping the BIT signature and a fault report cause model (curve) as shown in Figure 4.1.3-1.

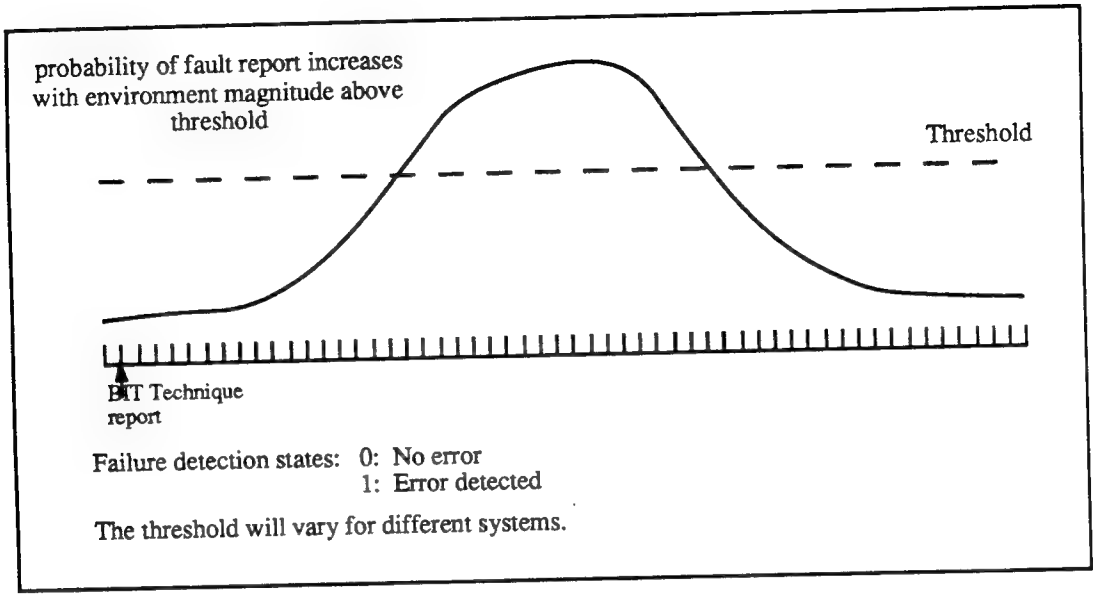


Figure 4.1.3-1. BIT Signature Combined With Fault Report Cause

System characteristics vary slightly based on the operational platform, operational life, and other situations. As a result, the failure threshold for a BIT circuit will vary for multiple versions of the same system and over time. Therefore, a boundary was defined, called the False Alarm/Intermittent Boundary, to represent the expected system threshold (see Figure 4.1.3-2).

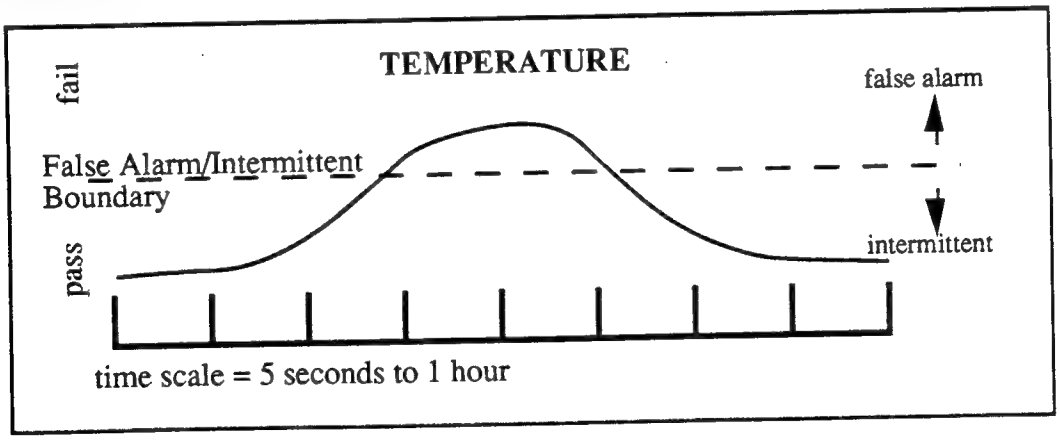


Figure 4.1.3-2. False Alarm/Intermittent Boundary

At different times, an actual system may have failure thresholds at different magnitudes than the boundary. If the threshold is above the boundary, then the actual system is operating better than the expected system and any failure reports will be false alarms. Conversely, if the threshold is

below the boundary, then the actual system begins to report failures before the boundary is reached. This represents an intermittent failure. As an example, Figure 4.1.3-3 shows a pass/fail BIT signature model with a temperature fault report cause.

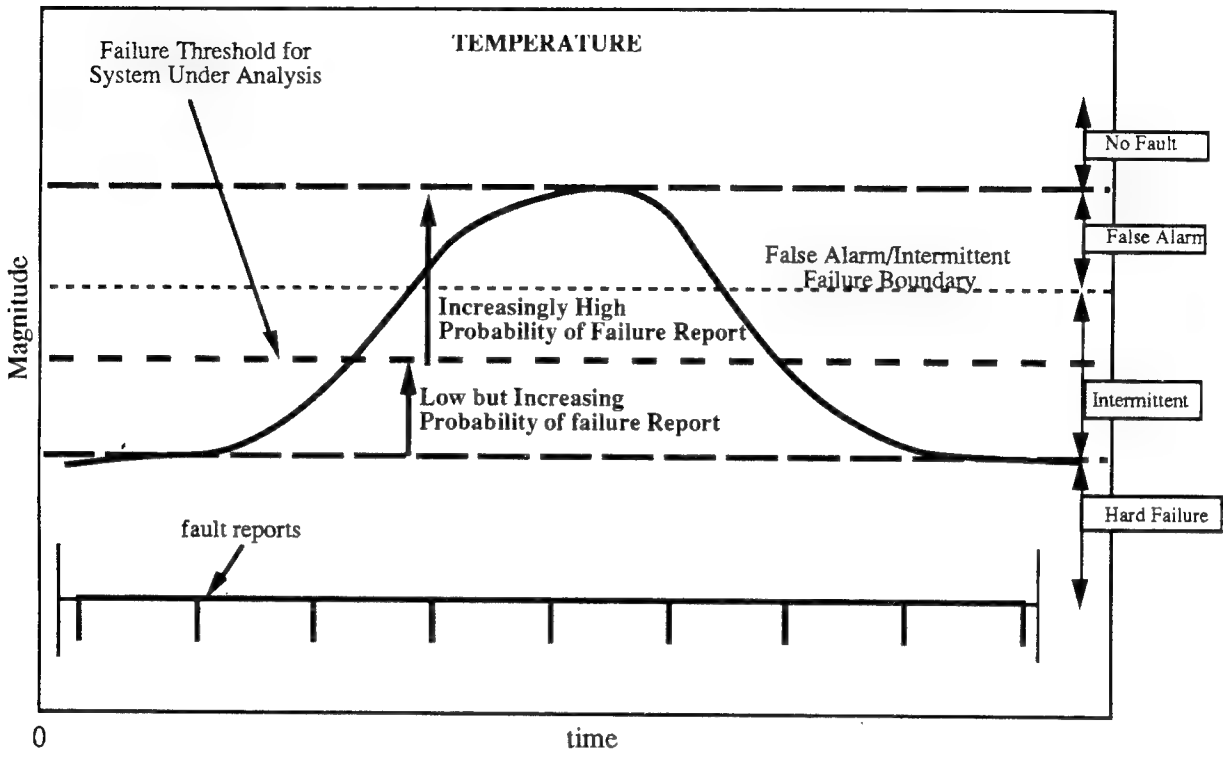


Figure 4.1.3-3 Pass/Fail BIT Signature with Temperature Fault Report Cause

This model represents a system that is susceptible to intermittent failures when the temperature exceeds the threshold. If the threshold is above the maximum magnitude of the fault report cause curve, then failures would not be expected and the system would operate effectively. If the threshold is at the zero magnitude of the fault report cause curve or lower, then the system would constantly report failures (the magnitude of the fault report cause would always be above the threshold) as would be expected from a hard failure.

From this model, network classifications were defined to correspond to the failure classifications described above. The No Fault class occurs when the threshold is at or above the maximum magnitude of the fault report curve. The False Alarm class occurs when the threshold is below the maximum magnitude of the fault report curve, but at or above the False Alarm/Intermittent Boundary. The Intermittent Failure class occurs when the threshold is below the False Alarm/Intermittent Boundary, but above the zero magnitude of the fault report cause curve. The Hard Fault class occurs when the threshold is at or below the zero magnitude of the fault report cause curve.

The combined BIT signature/fault report cause model produces a fault signature consisting of a stream of data bits over time. A pass/fail BIT fault signature will contain a binary data stream, whereas an error correcting BIT signature will contain a data stream with three values (0=pass, 1=corrected error, 2=non-correctable error). Examples of data streams for various thresholds, for

pass/fail and error correcting BIT signatures are shown in Figures 4.1.3-4 and 4.1.3-5, respectively.

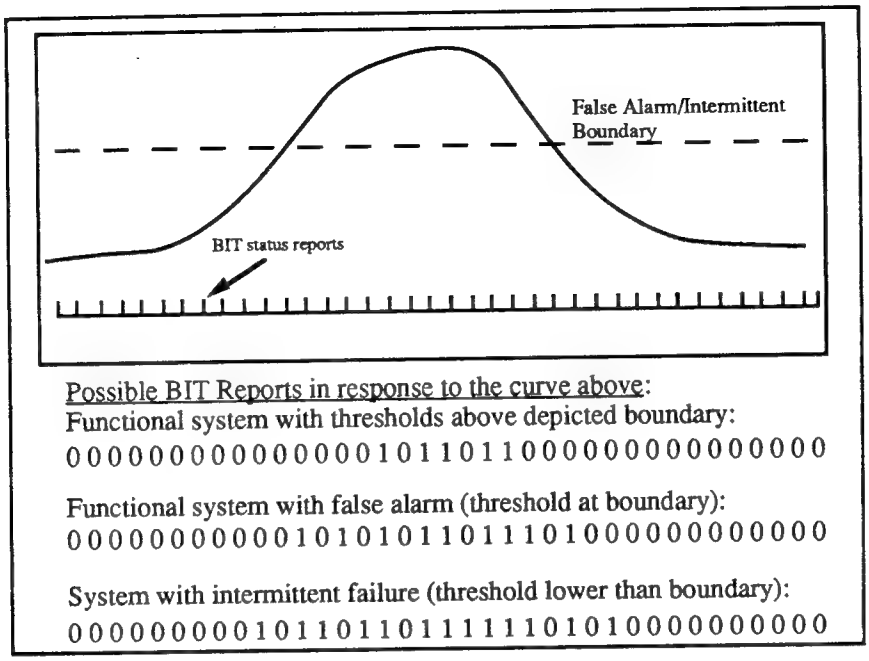


Figure 4.1.3-4. Data Streams for Pass/Fail BIT Signatures at Various Thresholds

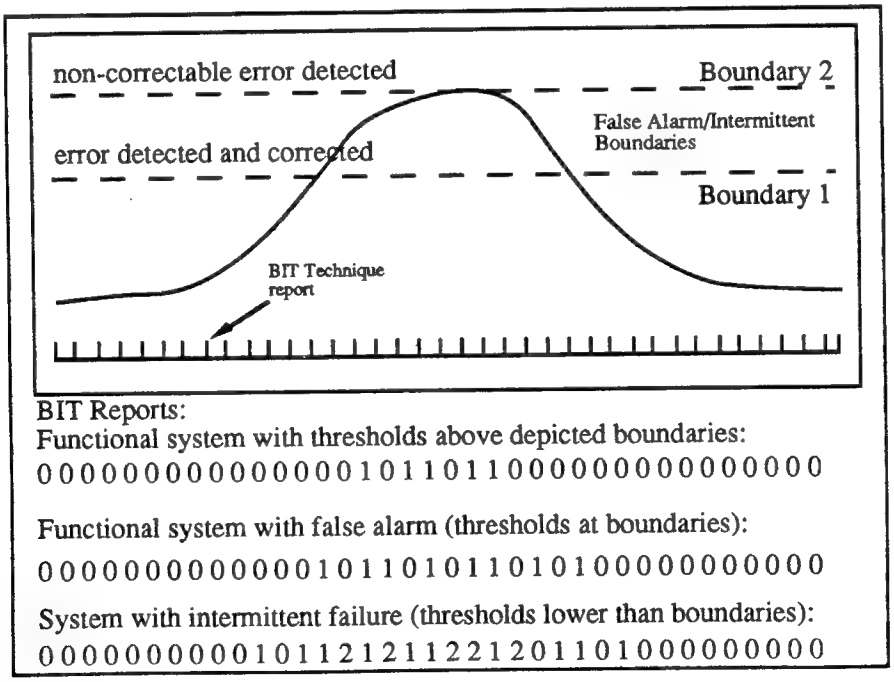


Figure 4.1.3-5. Data Streams for Error Correcting BIT Signatures at Various Thresholds

4.2 MILSTAR System

4.2.1 System Overview

The MILSTAR EHF/UHF terminal supports both a MILSTAR-specific frequency-hopping EHF/SHF waveform using sophisticated processing satellites, and a number of older existing UHF AFSATCOM modes employing existing non-processing satellites. Figure 4.2.1-1 is a simplified block diagram stressing the fault status collection reporting paths. A further description of the terminal and the bit collection/processing follows.

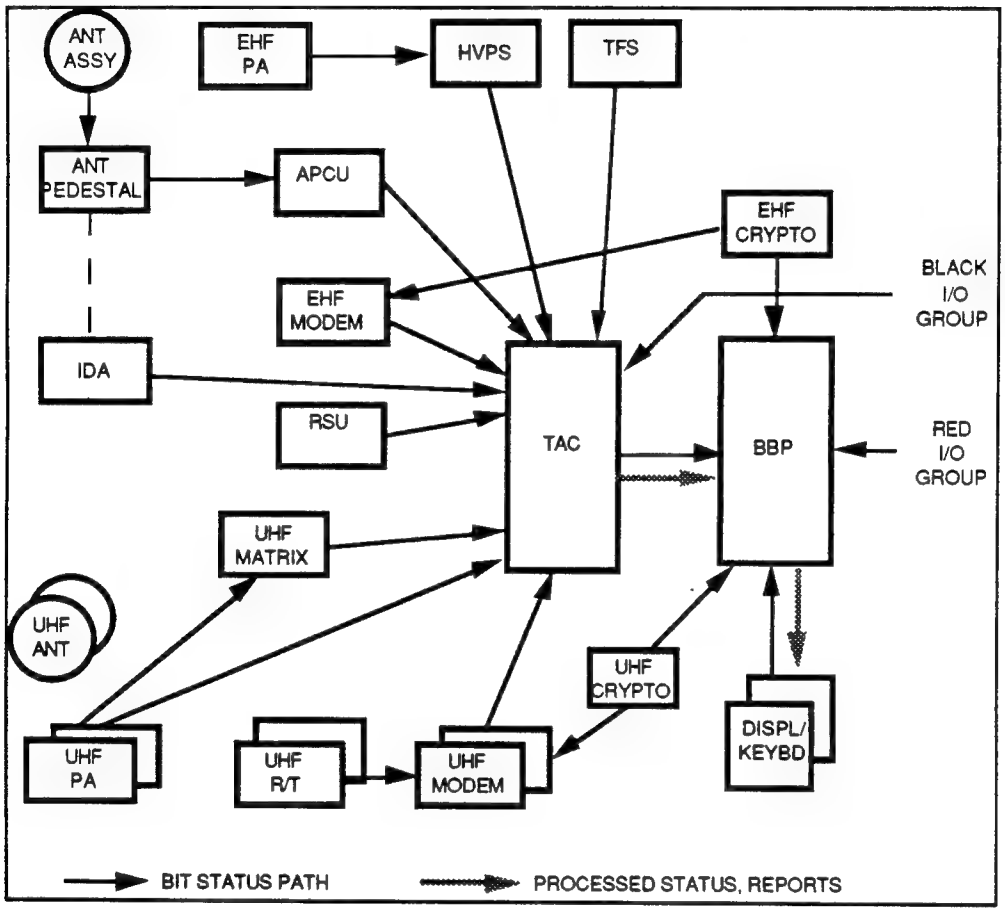


Figure 4.2.1-1. Simplified MILSTAR EHF/UHF Terminal Block Diagram

The terminal is functionally divided into three major groups: the common group, the EHF group and the UHF group. The common group provides functions common to MILSTAR EHF/SHF and AFSATCOM UHF modes, including general system control and status collection, operator/maintainer interfaces and frequency standards. The EHF group consists of the LRUs which provide EHF/SHF waveform modulation/transmission and receive/demodulation, power amplification , antenna control, and cryptographic functions. Sets of red (unencrypted) and black (encrypted or unclassified) user baseband ports are provided by dedicated interfaces within the common group. The UHF group provides essentially the same capabilities as the EHF group, except that it supports the backward-compatible AFSATCOM UHF waveforms and modes using predominantly existing inventory equipment.

The Common Group consists of the following major LRUs. The Terminal Access Controller (TAC) provides most of the control for the LRUs on the black side of the terminal, a suite of black baseband ports, and the satellite access protocols and tracking loops. It consists of a Data General-Eclipse based ROLM 16 bit processor, associated RAM, NVRAM and ROM, and a set of serial and parallel I/O ports.

The Baseband Processor (BBP) provides a suite of red baseband ports, associated multiplexing functions, control of the cryptographic subsystems, and interfaces to the operator display/keyboard(s). It is similar to the TAC, consisting of a Data General-Eclipse based ROLM 16 bit processor, associated RAM and ROM, a set of serial and parallel I/O ports, and a controller for the EHF cryptographic device which is hosted by the BBP.

The Time Frequency Standard (TFS) provides the basic timing references for the terminal system and maintains a time-of-day clock which will operate through nuclear event disruptions. It consists of a Cesium beam frequency standard and distribution amplifiers, time-of-day clock circuits and a microprocessor-based interface to the TAC .

The Display/Keyboard subsystem is actually two LRUs . They provide the operator input/output interface. The subsystem consists of a microprocessor-based plasma display and an enhanced keyboard. Terminals may have one or two display/keyboard systems, depending on the terminal type.

The EHF Equipment Group consists of the following major LRUs. The EHF Modem provides the MILSTAR uplink modulation and downlink demodulation functions, plays a part in satellite time, frequency, and spatial tracking, and supports black side cryptographic functions. The most complex of the MILSTAR LRUs, the Modem consists of 2 general purpose microprocessors, three 2900 bit-slice processors, multiple dedicated hardware cards, many which employ independent state sequencers, and an RF/analog section.

The Receiver/Synthesizer Unit (RSU) consists of two sets of ping-ponged synthesizers which provide hopped uplink and downlink Local Oscillator frequencies in response to tuning commands received from the EHF Modem. The RSU is implemented entirely with dedicated hardware, not only for the synthesizers but also for control and status functions.

The EHF Antenna Pointing and Control Unit (APCU) provides parts of the antenna pointing control loop and stabilization algorithms and amplifiers to drive the azimuth and elevation motors in the antenna assembly. The APCU is based on a microprocessor but also contains analog servo amplifiers and synchro resolvers, etc.

The Inertial Data Assembly (IDA) is a slightly modified existing inventory strap-down INS system mechanically coupled to the antenna pedestal. It provides three-axis heading information to the antenna pointing algorithms in the TAC.

The EHF Power Amplifier (EHFPA) and the High Voltage Power Supply (HVPS) are a pair of LRUs which provide final power amplification of the uplink EHF waveform using a traveling wave tube (TWT). The microprocessor-based HVPS interfaces to the TAC and provides the control functions and operating voltages for the EHFPA.

The EHF Antenna/Pedestal assembly provides the steerable antenna system for the uplink and the downlink signals. The complex includes a three-axis gyro assembly, a low noise amplifier/down-converter, a conical scan drive, and elevation and azimuth servos and synchros.

The UHF Equipment Group, not included with all terminal configurations, consists of the following major LRUs. The UHF Modems (2) provide waveform processing for most existing UHF SATCOM modes, plus adding a new Demand Assigned Multiple Access capability. The UHF Modems are comprised of microprocessor controllers and a variety of special purpose analog and digital cards.

The UHF Receiver/Transmitters (UHF R/T) (2) provide the RF portion of the UHF uplink and downlink chain. They are existing inventory ARC-121 units.

The UHF Power Amplifiers (UHFPA) (1 or 2) provide final power amplification of the UHF transmitted signals. Existing tube-type HPAs or newly designed solid-state HPAs are used depending on the platform type.

The UHF Matrix (UHFMA) provides a controller for a suite of relays which permit linking combinations of UHF R/Ts, Modems and HPAs, and tuning interfaces between the UHF Modems and the HPAs. The matrix is based on a microprocessor, a set of relay drivers and dedicated hardware to support the tuning interfaces.

4.2.2 MILSTAR Built-In Test

4.2.2.1 BIT Status Collection

The MILSTAR BIT is distributed among the LRUs. The LRU status is periodically collected and processed in the TAC and BBP to determine whether a condition exists which warrants a report to the operator or maintainer. Figure 4.2.2.1-1 shows an overview of the MILSTAR BIT status collection process, while each phase of the collection is described in more detail below.

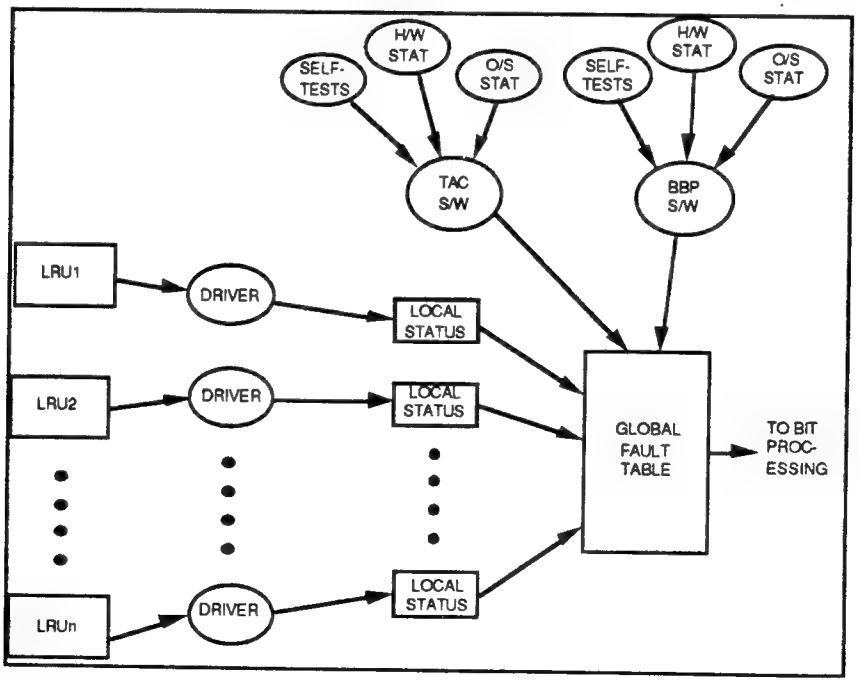


Figure 4.2.2.1-1. MILSTAR Terminal BIT Status Collection Flow

4.2.2.1.1 LRU BIT

Most LRUs contain independent BIT comprised of hardware-based detectors and firmware-based tests and analysis. All LRUs with microprocessors or processors execute a power-up test sequence, and in addition periodically sample and collect all their internal hardware fault detectors. Each LRU also monitors interface status to other LRUs including the serial control/status interface to the TAC or BBP. All fault status is accumulated and retained until sent to the TAC/BBP, at which time the internal LRU accumulation is cleared. In cases where there is no direct LRU status connection to the TAC/BBP, (e.g., the HPA or the antenna /pedestal assembly), the outlying LRU(s) will report status through another LRU which has a TAC or BBP status interface. The more sophisticated microprocessor-based LRUs such as the EHF Modem, the UHF Modems, the APCU and the TFS also provide a repertory of off-line tests which may be commanded over the control interface.

4.2.2.1.2 On-Line LRU Status Collection

Each major MILSTAR LRU connects to the TAC or BBP via a dedicated full duplex serial communications link called the System Control Interconnect (SCI). The SCI is used by the terminal software executing in the TAC/BBP to initialize and control the LRUs and receive operational and BIT status messages. In general, LRUs do not send unsolicited messages to the TAC/BBP; rather, fault status is "or" accumulated Internally until it is sent to the TAC/BBP in response to a received message. Some LRUs, such as the EHF Modem, provide preprocessing and reduction of the raw status prior to transmission. LRUs are polled for fault status at set intervals and whenever an operational message exchange is necessary.

In addition to fault information provided by the LRU, the interface driver for each LRU collects fault status from the TAC/BBP serial interface hardware (e.g., parity, overrun, break detect, etc.) and adds protocol-related status determined by software (e.g., incorrect message format or byte count, response time-out, etc.). The total status for each LRU, consisting of the LRU-generated status and the interface driver status, is "or" accumulated in a small table specific to each LRU. The update interval for the local LRU tables vary from 20 milliseconds to 1 second, depending on the message rate dictated by terminal operational needs.

The TAC and BBP software also collect hardware-based status internal to those LRUs and accumulate the status in local tables. Both processors also execute background diagnostics such as ROM check-sums and RAM tests and enter the results in their respective tables, along with critical status from the operating systems. In addition, a number of on-line loop tests and test channels provide continuous confirmation of the cryptographic subsystems and the EHF Modem (at both the digital and RF levels). Any failures noted by these tests are also accumulated in status tables.

Once every five seconds the LRU-specific tables are written into a global fault table (GFT), which contains a sub-space for all the local status associated with each LRU. At that time all local LRU tables are cleared and the GFT, which contains close to 500 bits of fault status, is passed to the BIT processing for evaluation.

4.2.2.1.3 Off-Line Testing and Status Collection

There are approximately 30 off-line tests which may be requested by the operator or maintainer, either individually or in groups of test sequences to exercise a particular subsystem of the terminal.

These tests include complete TAC and BBP self-tests, self-tests for each "smart" LRU, and a variety of loop tests which exercise a wider variety of paths than are possible on-line.

During and after the execution of each on-line test, status is collected in a manner similar to on-line, and the results written to a global off-line results table analogous to the GFT used on-line. Tests may self-terminate if interim status indicates that continuation of the current test or further tests in an automatic sequence may be invalid or hazardous. Test results and isolation messages are sent to the maintenance screens.

4.2.2.2 BIT Status Processing

The MILSTAR terminal processes the Global Fault Table data to determine whether a fault report is necessary, and to aid in fault isolation. The processing is table driven to facilitate upgrades during integration and as field experience is accumulated.

As shown in Figure 4.2.2.2-1, the BIT reduction and processing is table driven. Each of the 500 fault status bits contained in the GFT has a corresponding set of entries in the BIT Processing Table (BPT) which specify a fault precedence, values of N and M for discrimination, validation parameters (if applicable), a reporting severity parameter, certain isolation information, and pointers to a set of ASCII strings containing the appropriate screen and log messages if a fault report is generated.

Once each five seconds, fault status is collected from the local LRU tables into the GFT, where it is forwarded to the BIT processing. The processing of the GFT data is shown in Figure 4.2.2.2-1, and described below.

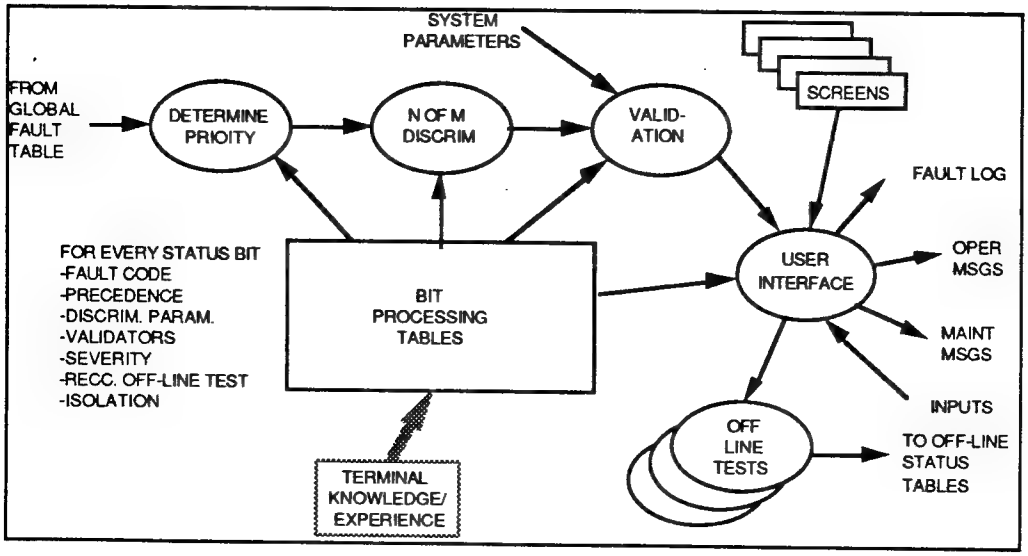


Figure 4.2.2.2-1. MILSTAR BIT Processing Flow

4.2.2.2.1 Fault Prioritization

The first phase of the processing scans the bits of the GFT in a hierarchical order according to the BPT, to determine the "highest" asserted fault symptom. The hierarchy is arranged based on two criteria. The first, and higher priority, criterion is to establish confidence in LRUs and functions

closest to the operator displays and BBP, then work progressively back into the system. Aside from assisting with isolation, this approach helps reduce false alarms by ignoring fault reports from lower hierarchy units if higher units are reporting symptoms which potentially corrupt the reporting mechanism. For example, the BBP is at the top of the hierarchy; if it is faulty then any other status reports in the GFT are suspect.

The second criterion is consideration of cases where a real failure in one hardware element propagates to multiple symptoms (collateral faults) potentially reported from multiple LRUs. An example of collateral fault propagation is a failure of the Time-Frequency Standard clock which causes activity and phase lock detectors to trip in multiple LRUs. For this reason the TFS is very high in the hierarchy.

In a few cases the two fault priority criteria conflict. These cases are handled by a "look ahead" escape in the BPT which permits conditional branching on specified GFT status bits lower in the hierarchy. For example, if the display serial port in the BBP is reporting a loss of reference clock, the BPT will specify looking ahead to certain TFS and EHF Modem faults to ascertain whether the problem is within the BBP or elsewhere in the system.

The output of the fault prioritization is a determination of the highest fault symptom for the most recent five second processing interval.

4.2.2.2.2 Fault Discrimination

The fault discrimination applies an N-of-M discriminator to the highest fault; that is, to pass discrimination the same highest fault must be present for at least N of the last M five second processing cycles. The discrimination processing will track N and M for up to twenty fault bits simultaneously. In the presence of multiple fault bits, the first fault to exceed discrimination is passed to the validation processing. The BPT can specify separate values for M and N for each of the 500 entries in the GFT.

4.2.2.2.3 Fault Validation

For each fault bit to which validation is applicable, the BPT specifies a set of operational metrics such as received signal level, bit error rate, time-, frequency-, and spatial tracking loop metrics as appropriate. If the designated parameters are within proper operational limits (despite the potential failure) then the severity of the report generated to the operator is downgraded. In some cases a fault symptom may be downgraded to the point where it will not be reported to the operator if, based on performance metrics, it is deemed to have no impact on mission performance. The fault data is always logged to the fault log and is available via the maintenance screens, however.

4.2.2.2.4 User Interface

The user interface processing generates fault report messages to the operator and maintainer. Faults are initially reported via the operational screens in one of three severity categories, according to data in the BPT: alarm, advisory, or log only. Since alarm messages are deemed urgent and invoke audible alarms and require explicit acknowledgment, they are reserved for fault reports which indicate a serious impact to terminal availability. Advisories inform the operator of potential degradation. The log-only category is reserved for symptoms which may require a maintenance action, but will not significantly impact mission operation. ASCII strings by the BPT provide fault isolation message strings and, where appropriate, a recommended off-line test to further evaluate a

problem. The user interface also provides a detailed set of maintainer screens and the ability to select and control off-line tests.

4.2.2.2.5 Fault Logging

In on-line mode, a fault message is sent to a logging file when ever there is a change in discriminated fault status, including the case where a fault may disappear. When running off line tests, the start and completion of a test and the resulting status is logged with no discrimination filtering. The log entries are time tagged and contain fault code and isolation strings. The log is automatically printed to hard copy by request or at terminal shut-down, or may be displayed in real time.

As described in the following section, for the purposes of generating simulated fault report data to train and test the neural networks, a logically related subset of the MILSTAR LRUs was selected, consisting of the TAC/BBP, the EHF Modem, the RSU and the HVPS/HPA. Note that for simulation purposes the TAC and BBP are considered one LRU because they are so similar. Likewise, the HVPS and HPA are considered one LRU because they are functionally highly interrelated even though they reside in separate chassis.

4.3 BIT Simulator

This section describes the BIT simulator used to generate the training and test data for the neural network models.

4.3.1 Simulation Alternatives

Two approaches were considered for the BIT simulator. The first approach would employ an existing MILSTAR simulator used by the software development facility to test and integrate MILSTAR software. The second approach would involve the development of a standalone simulator which would operate in a manner similar to the MILSTAR BIT processing.

The existing MILSTAR simulator is illustrated in Figure 4.3.1-1. The simulation runs on a Data General host computer, and the MILSTAR operational software runs in a test bed which contains a subset of the real MILSTAR hardware. The simulation is controlled by a scripted scenario which specifies the simulated bit patterns to be generated for each Line Replaceable Unit (LRU). The existing BIT scenario is capable of generating simulated faults for only one LRU. Recorded data is available for simulated LRU interfaces and for the global fault table used by the terminal BIT processing software.

The standalone simulator concept is depicted in Figure 4.3.1-2. In this approach, a single software component simulates a set of LRUs and BIT processing. The LRU simulations generate various fault signatures which are recorded in signature data files for processing and input to the neural network models.

Although the MILSTAR simulator was already in place, the existing BIT scenario for the simulator would have to be extensively modified to provide the multiple LRU fault correlation and the fault signature modeling capability required for the neural network models. The standalone approach offered the advantages of better asset availability, direct control over the output data, and greater flexibility through a tailor-made application. After analysis of both approaches, the standalone simulator approach was selected.

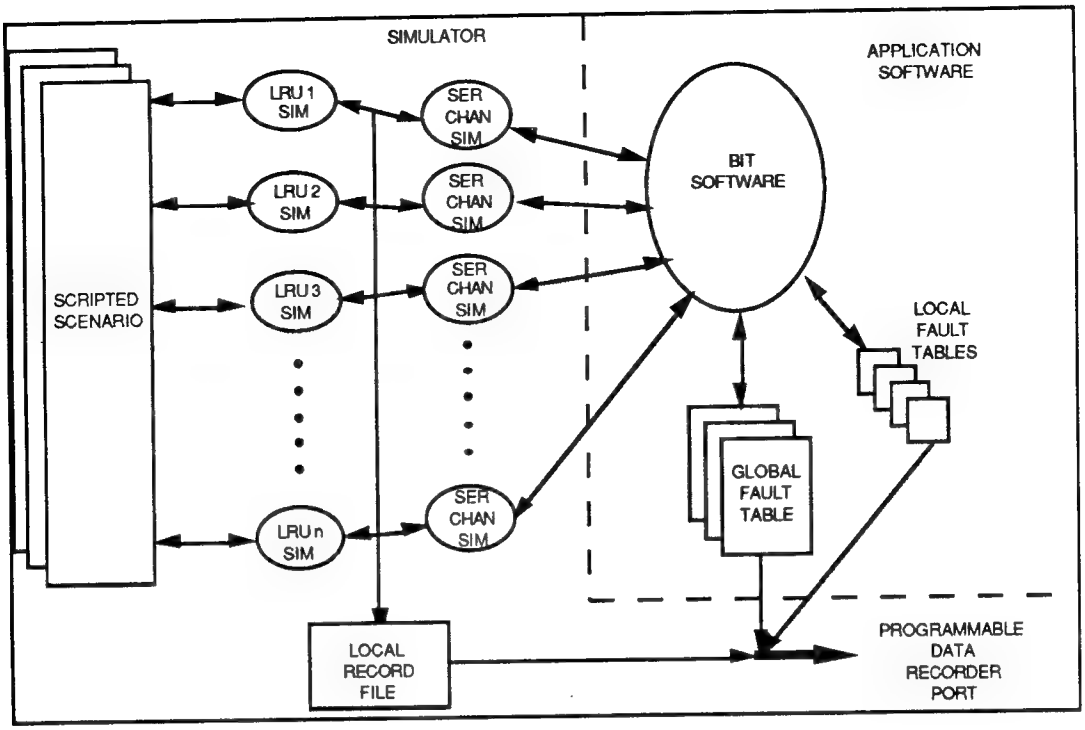


Figure 4.3.1-1. MILSTAR Simulator

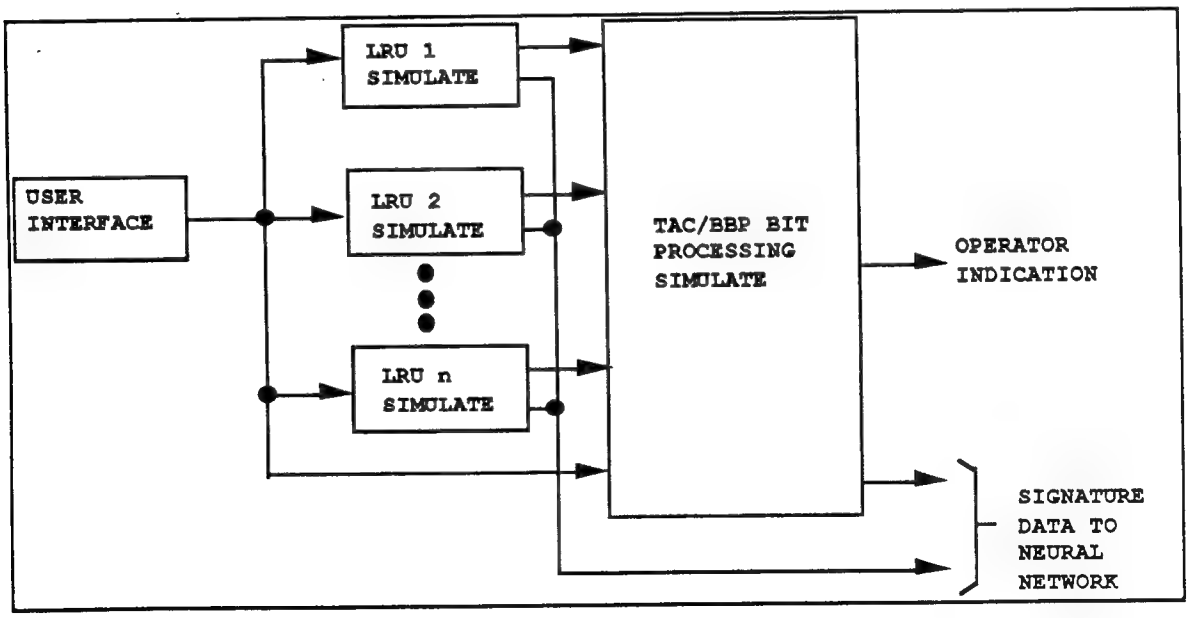


Figure 4.3.1-2. Standalone Simulator Concept

4.3.2 Simulator Description

4.3.2.1 Overview

The NNFAF BIT simulator is a software simulator based on the system BIT architecture of the MILSTAR satellite terminal. It is the component of the NNFAF software application which was used to generate fault signature data for training, testing, and validating the neural networks developed for the project. The simulator produces characteristic signatures for various FRC/BIT technique combinations. For each combination, signature data for false alarms, real faults, intermittents, and no faults can be produced. A block diagram of the NNFAF simulator is given in Figure 4.3.2.1-1.

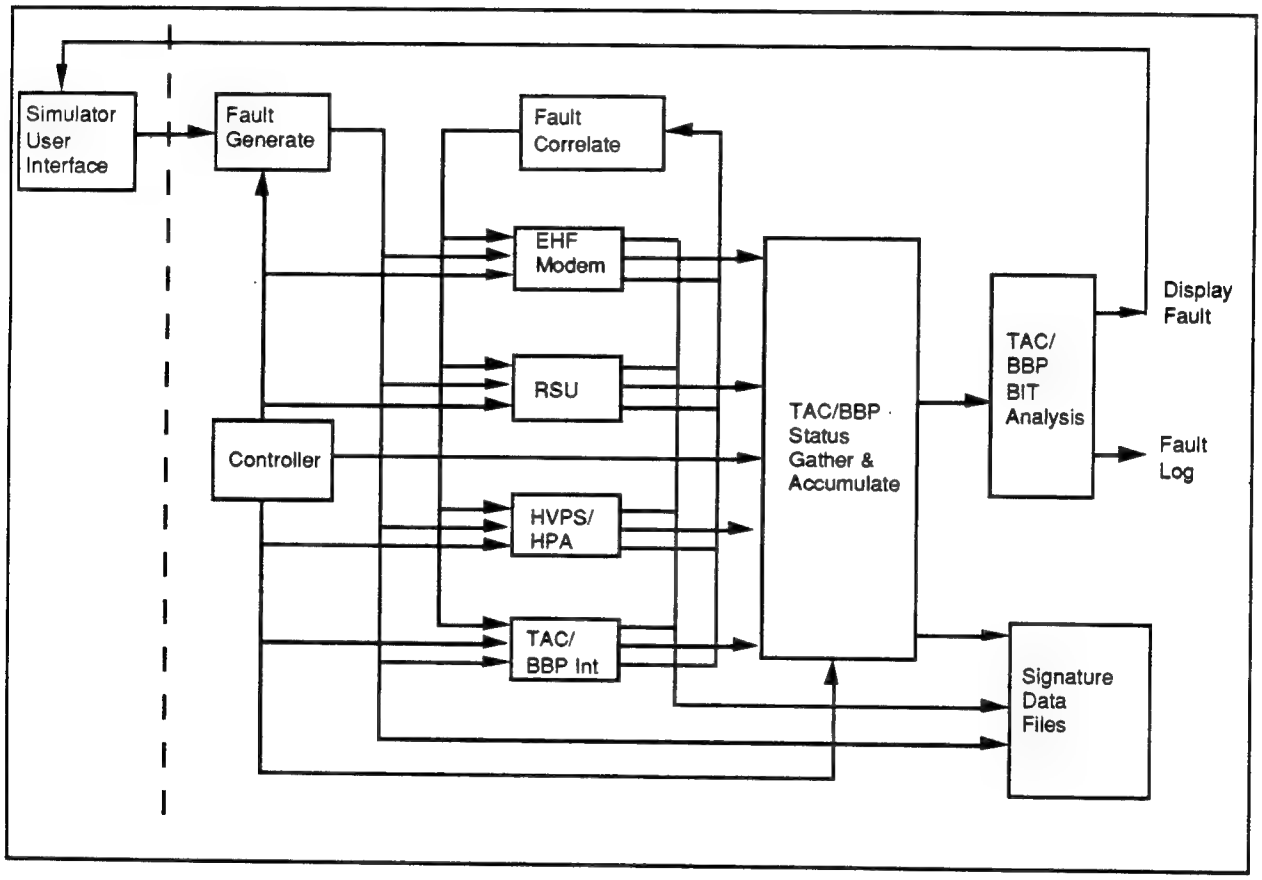


Figure 4.3.2.1-1. NNFAF Simulator Block Diagram

For the purposes of the NNFAF project, it was not necessary to simulate the entire MILSTAR terminal. Instead, the simulation was limited to four MILSTAR terminal LRUs. Together, the selected LRUs form a representative and logically interrelated subset of the MILSTAR terminal. The LRU selection process is described in detail in the following section.

4.3.2.2 LRU Selection

The goal of the LRU selection process was to select a representative subset of the MILSTAR LRUs which would be appropriate for the NNFAF demonstration. The selection process was based upon the following selection rules:

1. The chosen subset of LRUs should be logically interrelated in the real system so that interaction between faults can be realistically simulated.

2. The LRUs should represent all of the BIT status timing relationships used in the real system. At least one LRU should be selected from each of the following categories: Fixed-rate periodic status at the main BIT status cycle rate, fixed-rate periodic status at a higher rate than the main BIT status cycle rate, and random status rate determined by an event occurrence which is unrelated to the BIT status cycle.

3. The LRUs in the real system should actually implement the following BIT techniques chosen for simulation: convolutional encoding/viterbi decoding, activity detection, and parity generation/parity checking.

Based on the above rules, four LRUs were selected for the NNFAF BIT Simulator. Together, they perform most of the transmit and receive processing for the EHF portion of the real MILSTAR terminal. The LRUs are listed below, along with a description of the BIT cycle interval and the BIT technique applicability for each. The LRUs and their interrelationships are illustrated in Figure 4.3.2.2-1. The figure shows the roles of the units in actual transmit and receive message processing as well as the interfaces used for BIT processing. Only the BIT functions were implemented in the simulator.

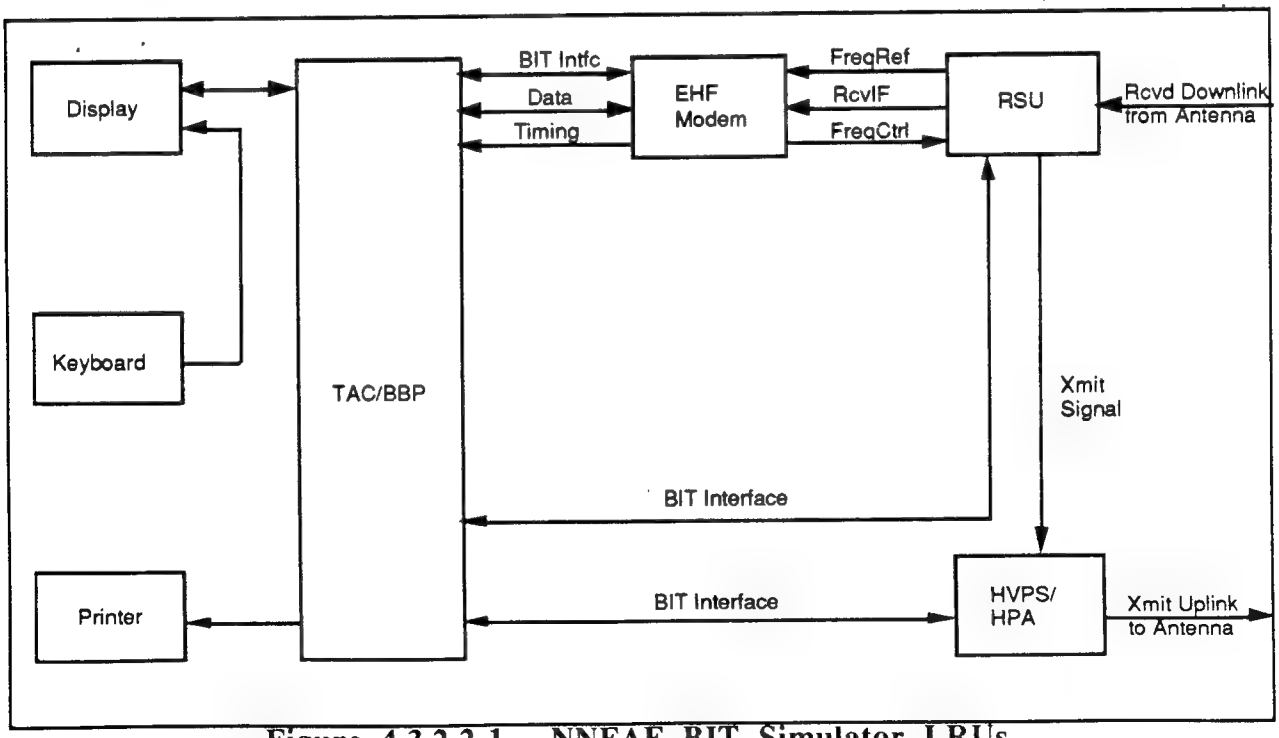


Figure 4.3.2.2-1. NNFAF BIT Simulator LRUs

1. Terminal Access Controller (TAC) and Baseband Processor (BBP). This pair of processors performs all of the software processing for the MILSTAR terminal, including terminal control, user interface, data routing, and BIT status collection and analysis. Since they operate as co-processors, they may be simulated by a single function. They contain a number of special-purpose input/output devices which are monitored and tested by BIT functions. BIT status is collected and accumulated each time a data transfer takes place through one of the devices in the normal course of operation. At the main BIT status cycle, the status of the input/output devices is collected and processed along with the all status for the terminal LRU hardware. BIT functions implemented in the TAC/BBP include activity detection and parity.

2. EHF Modem. This LRU converts data from baseband format into the MILSTAR transmit/receive waveform. It receives frequency references from the RSU, and it provides timing to the TAC/BBP and sends frequency control values to the RSU. It collects and accumulates BIT status internally at various cycle rates, then transfers the status to the TAC/BBP at the main BIT status cycle. The EHF Modem implements activity detection, parity, and encoding/decoding.

3. Receiver/Synthesizer Unit (RSU). Using frequency control values from the EHF Modem, the RSU synthesizes the frequencies used to generate the transmitted uplink signal to the High Voltage Power Supply and High Power Amplifier, and to downconvert the received downlink signal from the antenna. Once each main BIT status cycle, it collects BIT status from its internal hardware latches and transfers the status to the TAC/BBP. This LRU implements activity detection and parity.

4. High Voltage Power Supply (HVPS) and High Power Amplifier (HPA). The HVPS generates the voltages and control signals used in the HPA. The HPA performs the final up conversion of the uplink signal from the RSU and amplifies it for transmission. The HVPS collects BIT status for both itself and the HPA, and sends this to the TAC/BBP once per second. The TAC/BBP software accumulates this status until the main BIT status cycle, then it is cleared and the process is repeated. Since the HPA is slaved to the HVPS, and only the HVPS interfaces with the TAC/BBP, the two units can be simulated as one function. Of the selected BIT techniques, only parity is implemented by the HVPS.

4.3.2.3 Simulator Functions

The primary functions of the simulator are:

1. LRU Simulation, including FRC/BIT Technique Simulation
2. BIT Status Gathering and Accumulation
3. Fault Generation
4. Fault Signature Data Recording

Each of these is discussed in detail in the following sections.

4.3.3 LRU Simulation

4.3.3.1 EHF Modem Simulator

The EHF Modem simulator shown in Figure 4.3.3.1-1 simulates the fault status accumulation and reporting performed by the EHF Modem LRU. It includes the Temperature/Viterbi Fault Report Cause/BIT Technique simulator which generates the fault reports. Fault reports are written to a

latch, correlated with other system faults, and recorded in the signature data files. The latch holds any fault written to it until it is processed by the BIT processing function, at which time all faults in the latch are cleared. Inputs to the latch come from the Fault Report Cause/BIT Technique simulation and from the any correlated fault. When a BIT status request is received from the BIT status gathering and accumulation, the BIT processing function reads the latch, formats a BIT status report containing a single summary fault indication, and sends this report to the status gathering and accumulation mechanism.

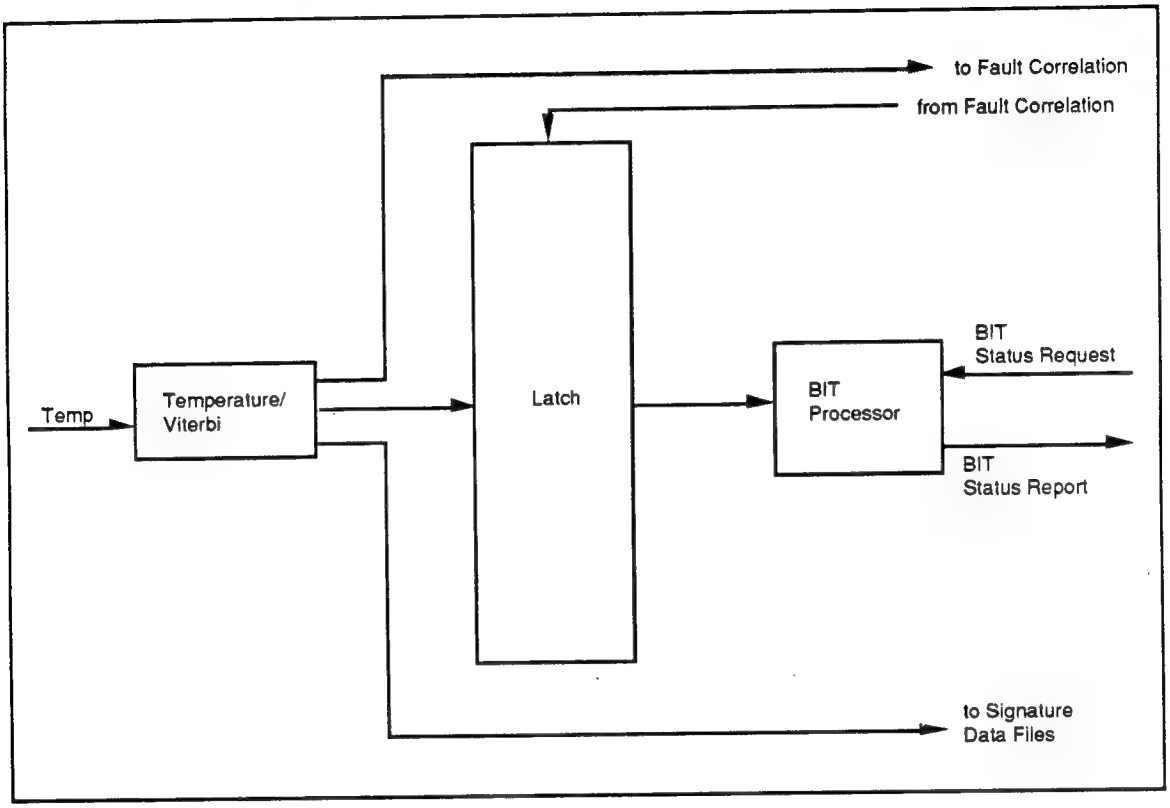


Figure 4.3.3.1-1 EHF Modem Simulator

4.3.3.1.1 Temperature/Viterbi FRC/BIT Technique Simulator

This simulator produces characteristic fault signatures for the effects of temperature variations on BIT techniques using convolutional encoding and Viterbi decoding on a data stream. A random data stream is produced at a rate representative of hardware. A standard 1/3 rate, 3 stage convolutional encoding is applied to the data stream. A probabilistic corruption is applied to the encoded data stream as a function of temperature and failure threshold. The corrupted data is decoded by the simulated Viterbi decoder, and fault indications are provided for both correctable and non-correctable faults. A diagram of this simulator is shown in Figure 4.3.3.1.1-1.

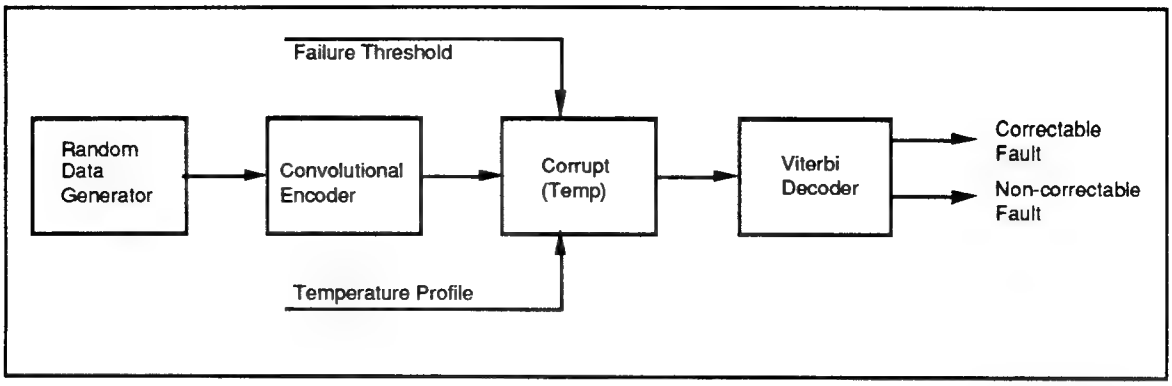


Figure 4.3.3.1.1-1. Temperature/Viterbi FRC/BIT Technique Simulator

4.3.3.2 RSU Simulator

The RSU simulator shown in Figure 4.3.3.2-1 simulates the fault status accumulation and reporting performed by the RSU LRU. It includes the Temperature/Activity Detector Fault Report Cause/BIT Technique simulator which generates the fault reports. Fault reports are written to a latch, correlated with other system faults, and recorded in the signature data files. The latch holds any fault written to it until it is processed by the BIT processing function, at which time all faults in the latch are cleared. Inputs to the latch come from the Fault Report Cause/BIT Technique simulation and from the any correlated fault. When a BIT status request is received from the BIT status gathering and accumulation, the BIT processing function reads the latch, formats a BIT status report containing a single summary fault indication, and sends this report to the status gathering and accumulation mechanism.

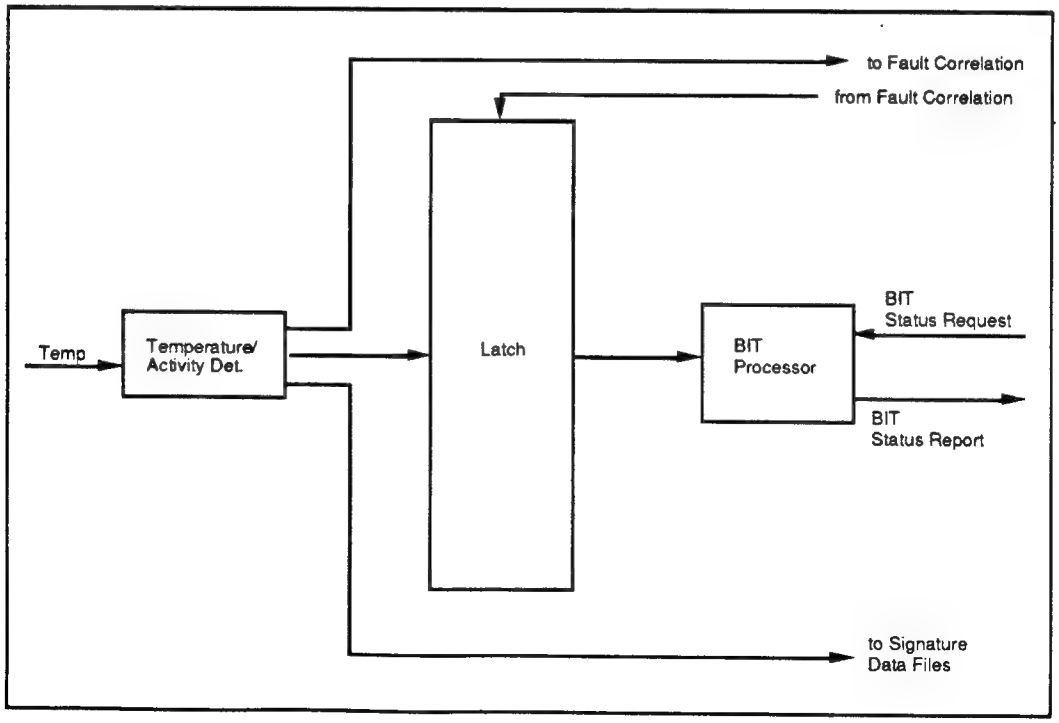


Figure 4.3.3.2-1. RSU Simulator

4.3.3.2.1 Temperature/Activity Detector FRC/BIT Technique Simulator

This simulator produces characteristic fault signatures for the effects of temperature variations on BIT techniques utilizing activity detectors. A random data stream is produced in which the data line transitions at least once in a representative time interval. A probabilistic corruption is applied to the data stream (the data is locked in one state for a period of time) as a function of temperature and failure threshold. A fault is produced if the data line fails to transition for a representative timeout period. A diagram of this simulator is shown in Figure 4.3.3.2.1-1.

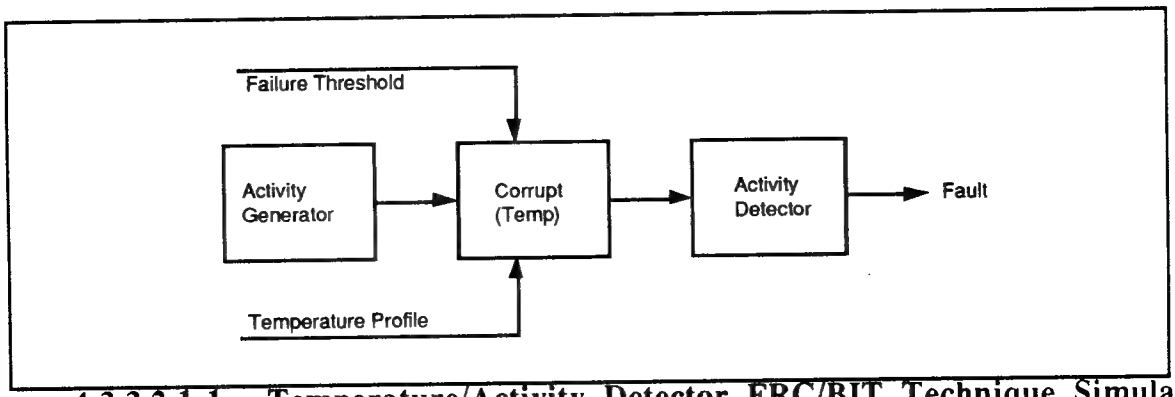


Figure 4.3.3.2.1-1. Temperature/Activity Detector FRC/BIT Technique Simulator

4.3.3.3 HVPS/HPA Simulator

The HVPS/HPA simulator shown in Figure 4.3.3.3-1 simulates the fault status accumulation and reporting performed by the HVPS/HPA LRU. It includes the G-Load/Parity Fault Report Cause/BIT Technique simulator which generates the fault reports. Fault reports are written to a latch, correlated with other system faults, and recorded in the signature data files. The latch holds any fault written to it until it is processed by the BIT processing function, at which time all faults in the latch are cleared. Inputs to the latch come from the Fault Report Cause/BIT Technique simulation and from the any correlated fault. When a BIT status request is received from the BIT status gathering and accumulation, the BIT processing function reads the latch, formats a BIT status report containing a single summary fault indication, sends this report to the status gathering and accumulation mechanism, and records it in the signature data file.

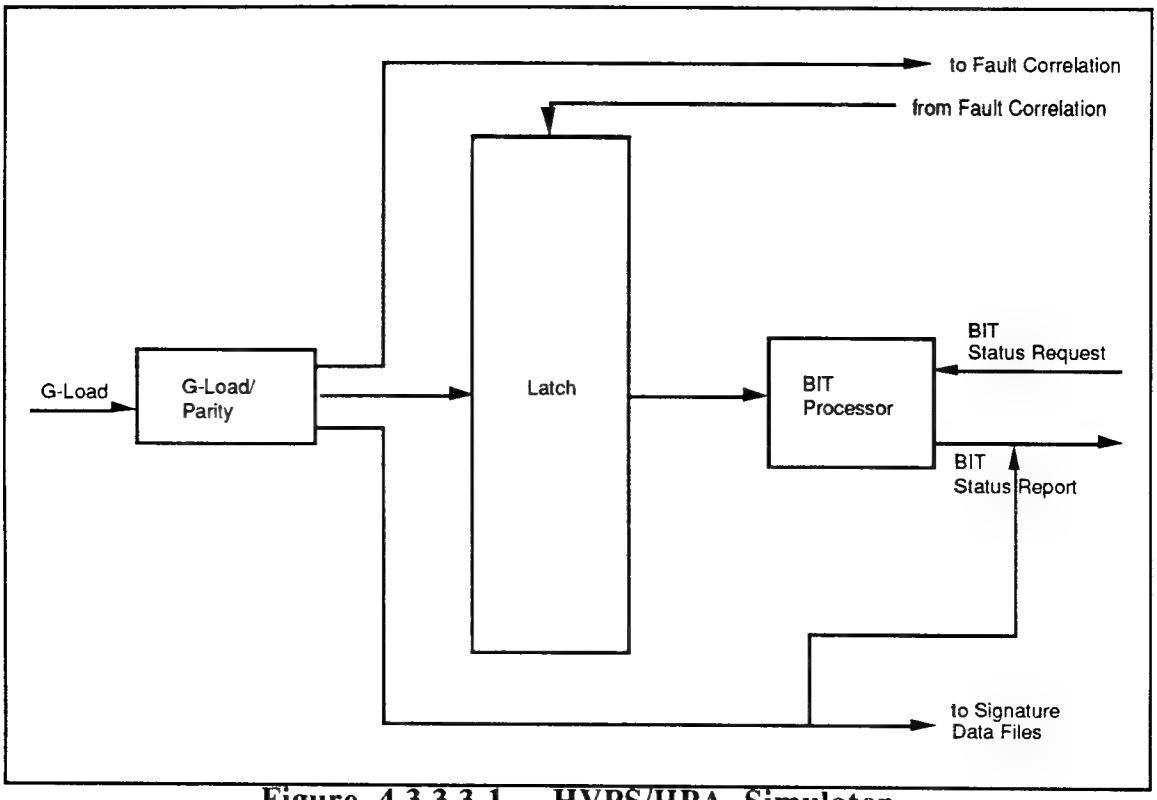


Figure 4.3.3.3-1. HVPS/HPA Simulator

4.3.3.3.1 G-Load/Parity FRC/BIT Technique Simulator

This simulator produces characteristic fault signatures for the effects of G-load on BIT techniques utilizing parity on a data stream. A random data stream is produced at a rate representative of hardware, and a parity bit is encoded onto each data word. A probabilistic corruption is applied to the encoded data stream as a function of G-load and failure threshold. The parity is recalculated on the corrupted data and a fault is produced when the parity check fails. A diagram of this simulator is shown in Figure 4.3.3.3.1-1.

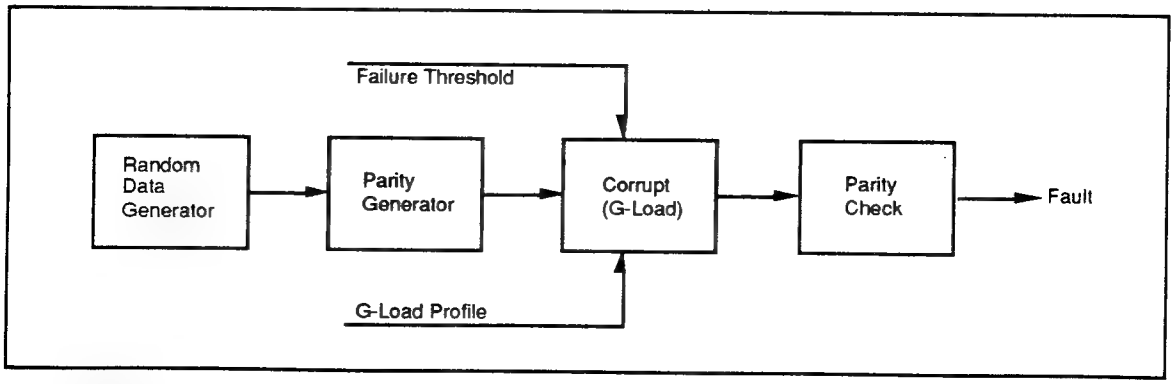


Figure 4.3.3.3.1-1. G-Load/Parity FRC/BIT Technique Simulator

4.3.3.4 TAC/BBP Internal Simulator

The TAC/BBP internal simulator shown in Figure 4.3.3.4-1 simulates the fault status accumulation and reporting performed by the TAC/BBP internal LRU. It includes the Vibration/Parity Fault Report Cause/BIT Technique simulator which generates the fault reports. Fault reports are written to a latch, correlated with other system faults, and recorded in the signature data files. The latch holds any fault written to it until it is processed by the BIT processing function, at which time all faults in the latch are cleared. Inputs to the latch come from the Fault Report Cause/BIT Technique simulation and from any correlated fault.

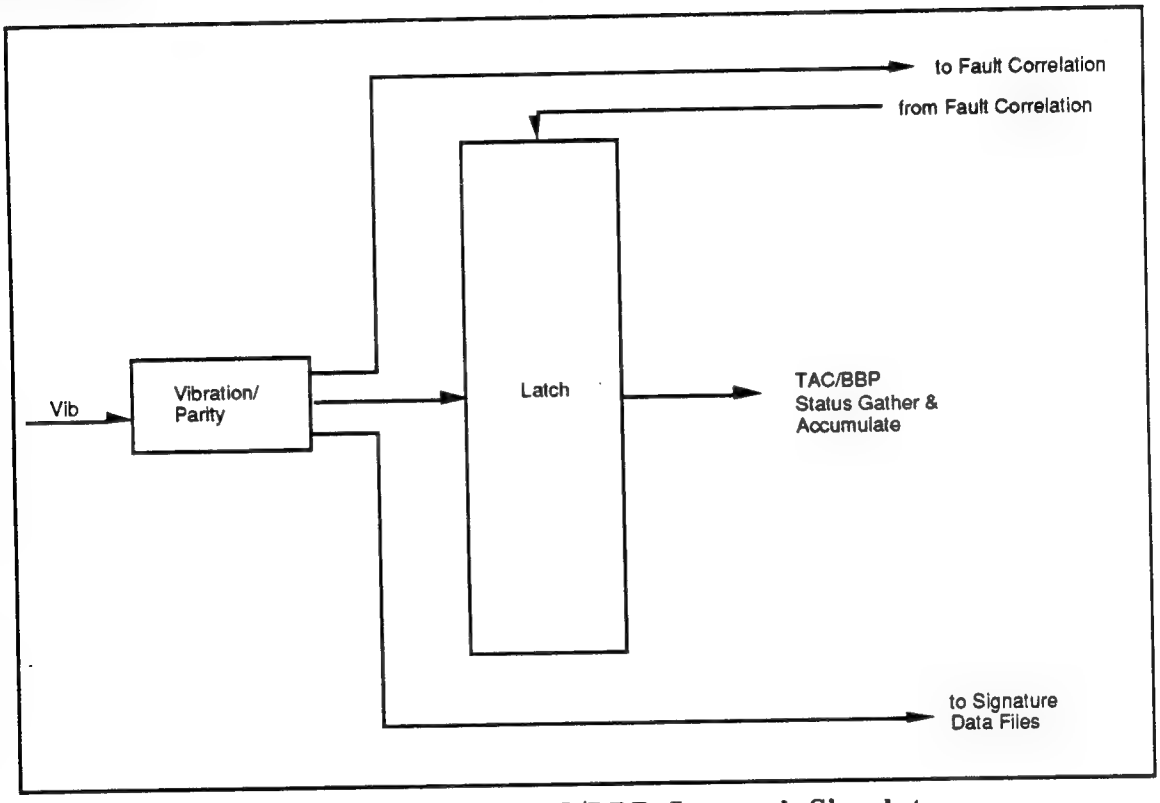


Figure 4.3.3.4-1. TAC/BBP Internal Simulator

4.3.3.4.1 Vibration/Parity FRC/BIT Technique Simulator

This simulator produces characteristic fault signatures for the effects of vibration on BIT techniques utilizing parity on a data stream. A random data stream is produced at a rate representative of hardware, and a parity bit is encoded onto each data word. A probabilistic corruption is applied to the encoded data stream as a function of vibration and failure threshold. The parity is recalculated on the corrupted data and a fault is produced when the parity check fails. A diagram of this simulator is shown in Figure 4.3.3.4.1-1.

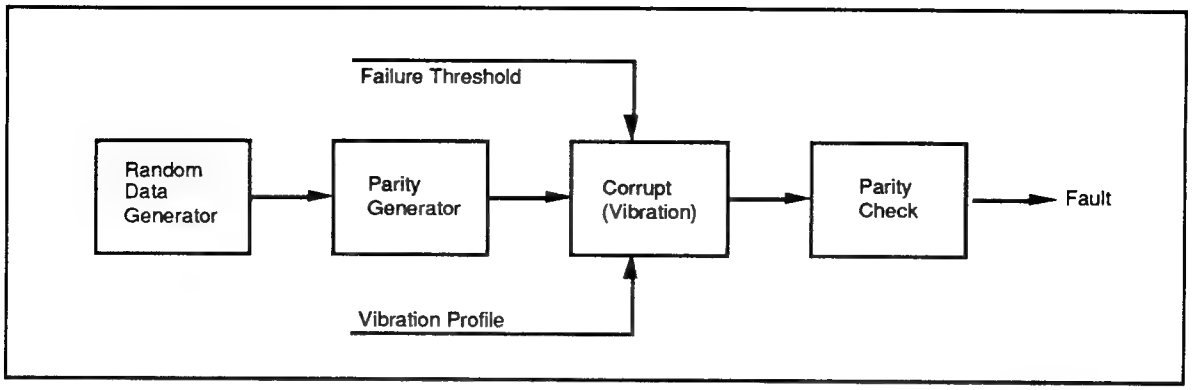


Figure 4.3.3.4.1-1. Vibration/Parity FRC/BIT Technique Simulator

4.3.4 BIT Status Collection and Accumulation

This component simulates the action of the TAC/BBP software in gathering BIT status from the individual LRUs and accumulating and storing the status into a single Global Fault Table (GFT) for analysis (see Figure 4.3.4-1). There are two BIT status cycles which are simulated. The main BIT status cycle is simulated to occur once every 5 seconds, and this parameter may be varied by the user. The HVPS/HPA status cycle is an intermediate status gathering point, and is simulated to occur once every second. Once each HVPS/HPA status cycle, the HVPS/HPA status is requested and stored into a latch. Once each main BIT Status cycle, the HVPS/HPA latch is read and stored into the GFT, and status from the other three LRUs is requested and stored into the GFT. If the main BIT status cycle rate is less than the HVPS/HPA status cycle rate, the HVPS/HPA status cycle rate becomes the main BIT status cycle rate.

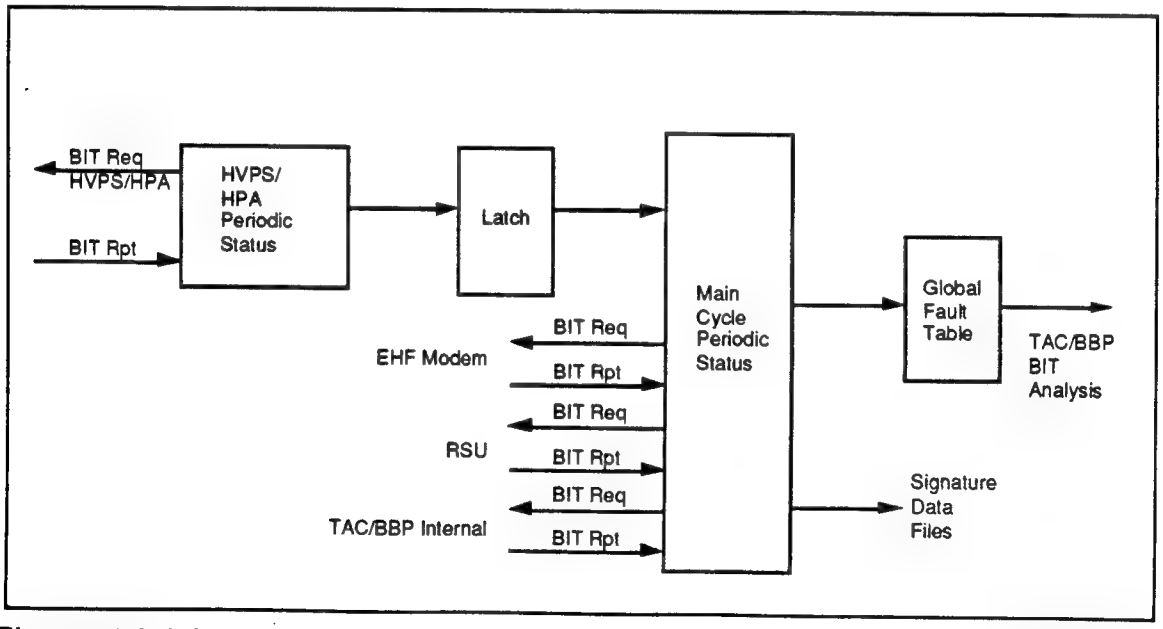


Figure 4.3.4-1. NNFAF Simulator BIT Status Collection and Accumulation

4.3.5 Simulated Fault Generation

The primary functions of the each of the FRC/BIT Technique simulators are to simulate the generation of faults and to record fault signatures for use by the neural network models. Each run of the simulator consists of the generation of one or more fault signatures, based on an environmental profile and a set of user-selectable options. These options determine the characteristics of the simulation, including the shape and degree of random perturbation of the environmental profile, the level of the environmental input which will begin to produce faults, and the number of signatures that will be generated. Four options are available for failure threshold determination. These options allow the failure threshold to be varied automatically over many signatures in the run. The simulator options are described in detail in section 4.3.5.1.

4.3.5.1 Simulator Options

1. Simulation Approach. This option provides for the selection of the simulation approach. The approaches are Temperature/Viterbi, Temperature/Activity Detect, G-Load/Parity, and Vibration/Parity. Based on this selection, the appropriate LRU simulation is performed.

2. Environmental Profile Curve. The specific environmental profile can be selected from a list of environmental profile curve files. Only the curve files which pertain to the selected approach are displayed. A curve file contains environmental profile data consisting of normalized time (x axis) and normalized magnitude (y axis) values over a default period of time.

3. Environmental Profile Curve Length. This parameter is a scale factor which, if selected, is applied to the time axis of the selected environmental profile curve. It determines the curve duration.

4. Mean Perturbation. This parameter is used as the mean of a Gaussian noise distribution which is to be applied to the magnitude of the selected curve. It represents the average difference between the specified curve and the noise applied to the curve.

5. Standard Deviation. This parameter is used as the standard deviation of a Gaussian noise distribution which is to be applied to the magnitude of the selected curve.

6. Failure Threshold. This parameter represents the threshold of the environmental input at which faults will begin to be produced.

7. Fixed Threshold. If this threshold option is chosen, the user specifies a single failure threshold value and a fixed number of signatures to be generated.

8. Fixed Delta. If this threshold option is chosen, the user specifies a lower bound, a delta value, and an upper bound for the threshold value. Starting from the lower bound, one signature is generated for each delta until the upper bound is reached.

9. Threshold Array. If this threshold option is chosen, the user enters the desired number of threshold values, that number of threshold values, and the desired number of signatures. The specified number of signatures will be generated at each threshold value.

10. Random Thresholds. If this threshold option is chosen, the user specifies a lower bound, an upper bound, and a number of signatures. For each signature, a threshold value is generated with a random value between the lower and upper bound.

11. False Alarm/Intermittent Boundary. This parameter represents the boundary for levels which will be used to classify fault signatures.

12. System Functional with Reporting Malfunction. If this parameter is selected, it is used to indicate that a fault report is occurring even though the system is fully functional. When this parameter is selected, all faults are classified as "BIT Only", and no correlated faults are generated.

13. Timing Options. There are two timing options: Main BIT Status Cycle Rate, and Clock Granularity. The Main BIT Status Cycle Rate is used to modify the main status gathering and accumulation time period. The clock granularity is used to modify the periodicity of the simulator itself.

14.. Seed Random Number Generator. This parameter selects a seed for the random number generator. This parameter is used to assure both replicability and uniqueness of experiments.

4.3.5.2 Fault Signature Data

Fault signature data is recorded into four distinct data files. One file contains fault data from the source of the fault, one file contains data from the intermediate (HVPS/HPA or LRU) status level, one file contains data from the main BIT status (GFT) level, and one file contains a combination of all fault bits from the GFT. The HVPS/HPA is the only LRU that has an intermediate status level, so this file is primarily of interest to the G-Load/Parity approach. However, the HVPS/HPA status may potentially contain a correlation fault for another approach. For this reason it is stored for all approaches. The intermediate status level can be simulated for the other approaches by shortening the main BIT status cycle rate. Figure 4.3.5.2-1 shows the flow of data to the various signature data files.

Each entry in a fault signature data file contains a time tag, the fault indication, and the value of the appropriate environmental data at that point in simulated time. Examples of each level of fault signature data files can be found in Appendix G.

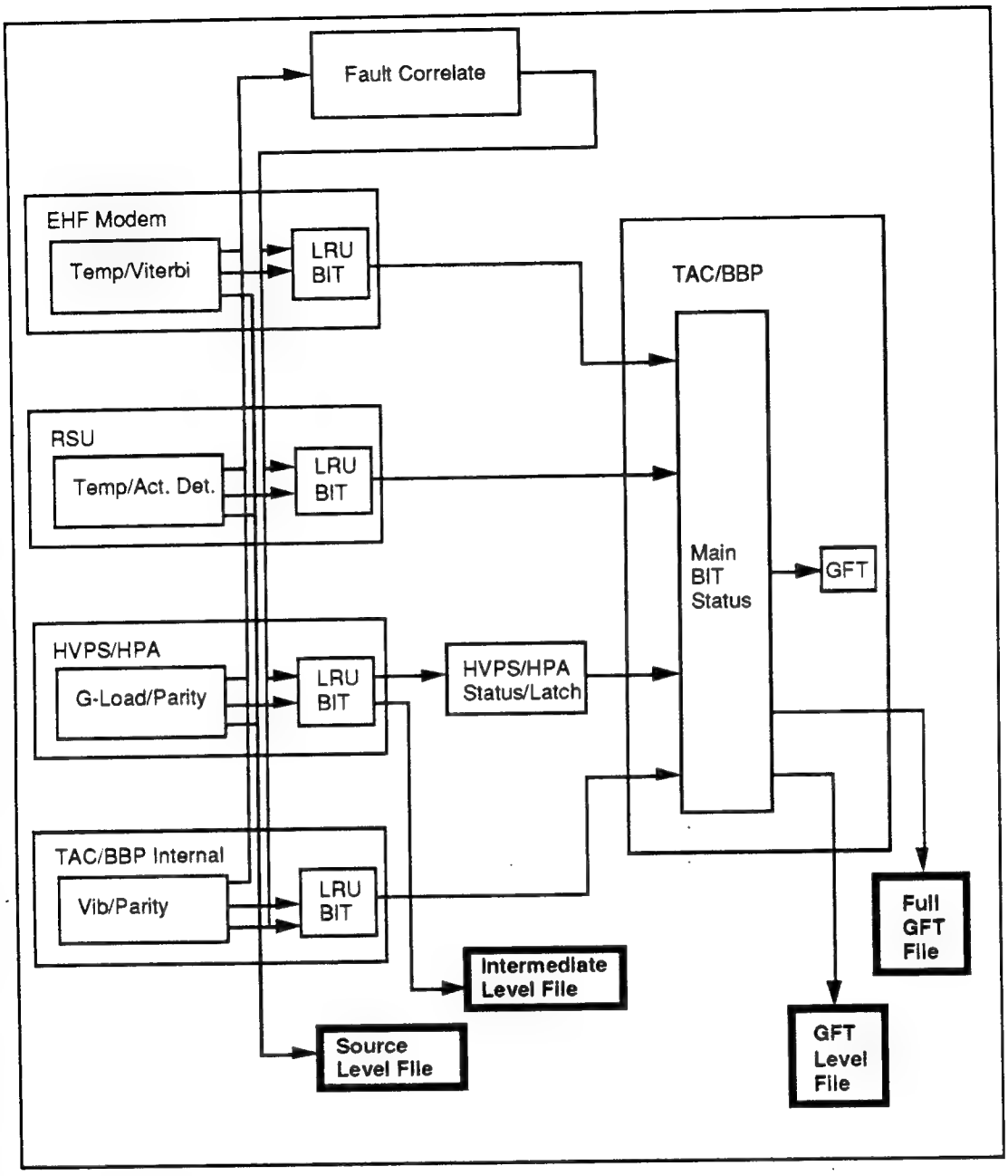


Figure 4.3.5.2-1. NNFAF Fault Signature Data Recording

5. DEVELOPMENT METHODOLOGY

5.1 Hardware Platform

The hardware platform for the NNFAF demonstration is as shown in Figure 5.1-1. The simulation software was developed on a Personal Computer (PC) 386, in the ANSI standard C language. This platform was selected because of its availability to the software developer. The

neural network, user interface, and all other support software was developed on a Sun IPX platform, with the configuration shown in the figure. The Sun was selected as the preferred platform for the majority of the software development because it was to be the final demonstration platform at the customer site. The simulation software was ported to the Sun platform after its unit test was complete. The final delivery and demonstration platform was the Sun Workstation at Rome Laboratory.

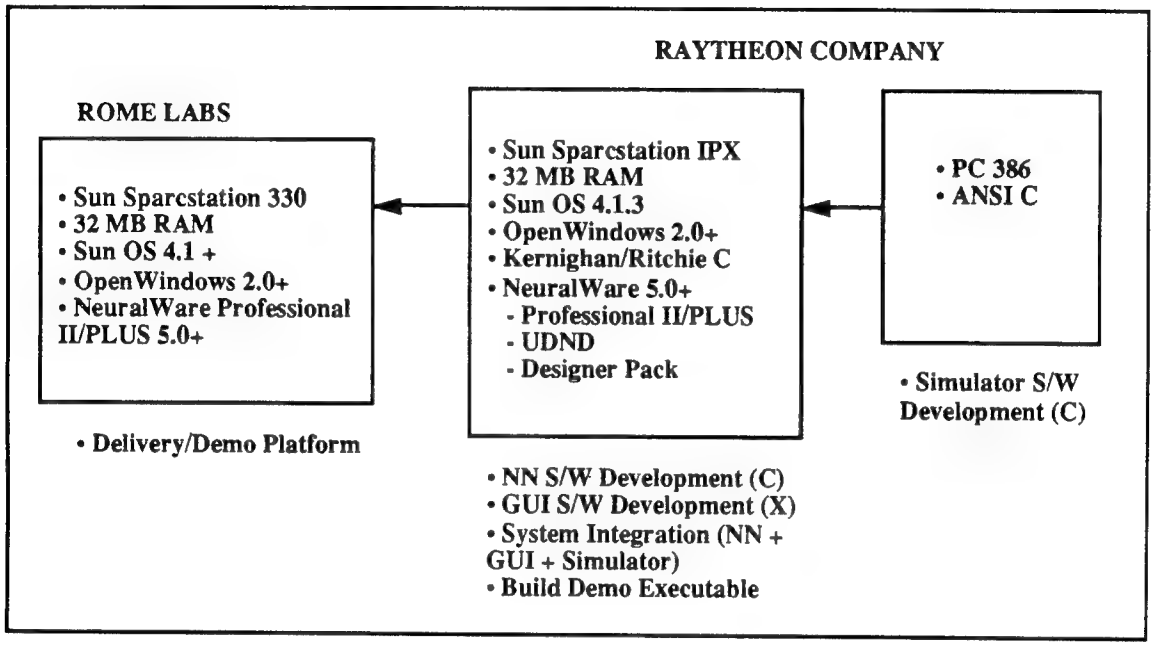


Figure 5.1-1. NNFAF Development and Delivery Hardware Platforms

5.2 Software and Software Tools

The NNFAF software consists of the following items:

- the Commercial Off-The-Shelf (COTS) software used to develop the neural networks;
- the custom software developed to provide a graphical user interface, a data simulator, and neural network modeling and data processing capability.

The following sections describe these items in more detail.

5.2.1 NeuralWare Software

The COTS software used for the NNFAF project consisted of the neural network development software by NeuralWare. This software is made up of three distinct products:

- NeuralWorks Professional II/PLUS which is the main neural network development tool. It allows the user to build, train, refine, and deploy neural networks.
- NeuralWorks User Defined Neuro Dynamics, which allows the user to create custom-designed neural network architectures within the framework of the NeuralWorks environment.
- NeuralWorks Designer Pack, which provides a neural network deployment capability.

5.2.2 Custom Software

The NNFAF custom software consists of the Human-Machine Interface, the Simulation Software, and the Neural Network Modeling and Data Processing Software. The software system block diagram is shown in Figure 5.2.2-1.

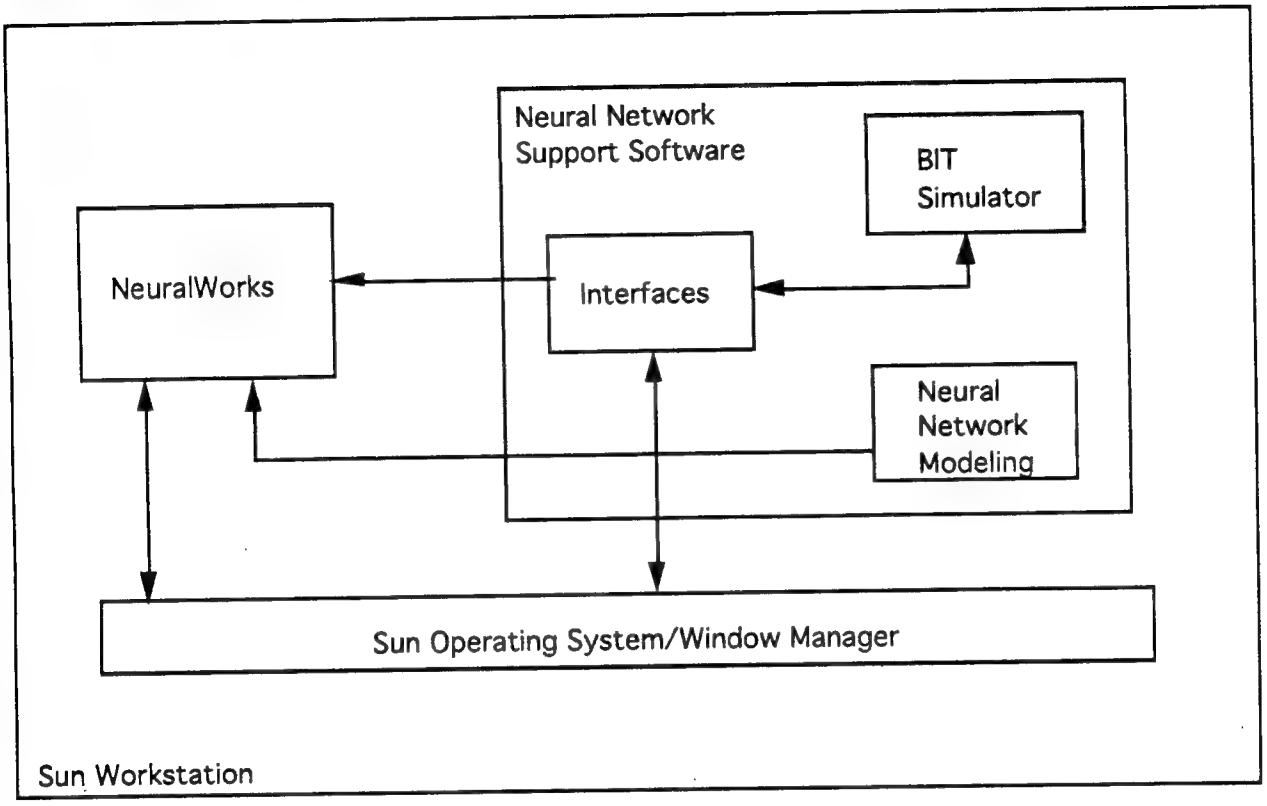


Figure 5.2.2-1. NNFAF Software System Block Diagram

5.2.2.1 Human-Machine Interface

The Human-Machine Interface, also called the Graphical User Interface (GUI) is an X-windows application. It was written using the Sun OpenLook Intrinsics Toolkit, in the C language. It consists of a main menu with submenus for executing the NNFAF simulator and the NNFAF demonstrations. The NNFAF Software Design Document and Software Users Manual contain detailed descriptions of the NNFAF GUI and representations of the NNFAF windows. The basic function of the GUI software are to display software identification information, to receive and process user input, and to present selection options which allow the user to execute either the simulator or the neural network demonstrations.

The windowing environment of the NNFAF demonstration is a multi-window environment. At any time, there may be multiple windows present on the display, including one or more console or shell windows for Unix access, windows for Sun Workspace functions, the NNFAF windows, and/or the NeuralWorks windows. The user has unrestricted access to any of these windows at any time.

5.2.2.2 Simulation Software

The functions of the simulation software have been described in Section 4.3. The software was written in the C language. It is described in detail in the NNFAF Software Design Document and Software Users Manual.

5.2.2.3 Neural Network Modeling and Data Processing Software

The neural network modeling software was developed in the C language to implement neural network capability not provided by the NeuralWorks products. The neural network data processing software was developed in the C language to provide formatting, analysis and plotting capability for the neural network input and output data.

The data processing software is used to format the fault signature data output from the simulator. Each neural network model requires that the training and testing data be in a certain format. The data processing software manipulates the signature data so that it is compatible with the given neural network model. There are three data formatters, one for each type of neural network in the NNFAF demonstration system. In addition to the input data formatters, software was also developed to format and present the network results.

This software is provided off-line, and is not accessible through the NNFAF GUI. It is documented in the NNFAF Software Design Document and Software Users Manual.

5.3 Software Development Methodology

The Neural Network False Alarm Filter software was developed using a tailored DOD-STD-2167A (2167A) approach. The Statement of Work tailored 2167A such that only the following paragraphs were applicable:

- Software Development Management, paragraphs 4.1, 4.1.1 (a-d) only;
- Software Engineering, paragraph 4.2, excluding paragraphs 4.2.2, 4.2.3, 4.2.5, 4.2.6, 4.2.8, 4.2.9, 4.2.10;
- Preliminary Design, paragraph 5.3, excluding paragraphs 5.3.1, 5.3.2.4, 5.3.3, 5.3.4, 5.3.5;
- Detailed Design, paragraph 5.4, excluding paragraphs 5.4.1, 5.4.2.4, 5.4.2.5, 5.4.3, 5.4.4, 5.4.5.

A non-deliverable, informal Software Development Plan (SDP) was written to define the software process for the NNFAF program. A non-deliverable, informal Software Requirements Specification (SRS) was also written to define the requirements for the software. The Preliminary Design portion of the Software Design Document (SDD) was completed for the user interface and simulator portions of the software. The detailed design for the software was completed in increments. A version of the Simulation Software was designed and coded, containing all basic functionality. A second version was then designed and coded, including additional capabilities. The Interface Software was initially designed and coded, but was expected to evolve throughout the life of the program. The neural network data processing software was also developed in increments to accompany the incremental development of the neural networks.

A draft Software User's Manual (SUM) and draft R&D Test Plan (Demo Plan) were developed during the Preliminary Design phase of software development. The SUM and Demo Plan were updated throughout the life of the program.

5.4 Neural Network Development Methodology

The general neural network development methodology consisted of 5 steps:

1. Understand the simulated data, including its format, content, and how to generate it using the various simulator parameters.
2. Format the simulated data into training, testing, validation, and noise testing files for input to the network.
3. Construct, optimize, train and test the network using NeuralWorks.
4. Perform validation and noise testing using NeuralWorks.
5. Collect and analyze network results.

For each approach, one or more networks were developed and exercised. Individual methodologies for each approach are discussed in the individual sections that follow.

Validation data is a term used by NeuralWare, Inc. to identify a neural network input data set which the network has never seen before. Validation data is distinguished from training or testing data, both of which are repeatedly shown to the network during network training and optimization. NeuralWare's terminology for the different data sets was adopted for this project. Note that the validation data was used as the acid test of network performance only. Validation did not entail validation of the fault model or simulation concepts.

5.4.1 Approach 1: Backpropagation/G-load/Parity

Approach 1 applied the Backpropagation neural network model to the Parity BIT technique, affected by a G-load fault report cause. Figure 5.4.1-1 shows a visualization of the simulated G-load event, overlaid with an example of a fault report signature for this approach. The duration of the G-load event was defined to be 1 minute (60000 msec) and a corresponding G-load environmental curve file was generated (see Appendix F). The parity BIT simulator for this approach is described in section 4.3.3.3.1. It was designed to produce a BIT report every 200 msec.

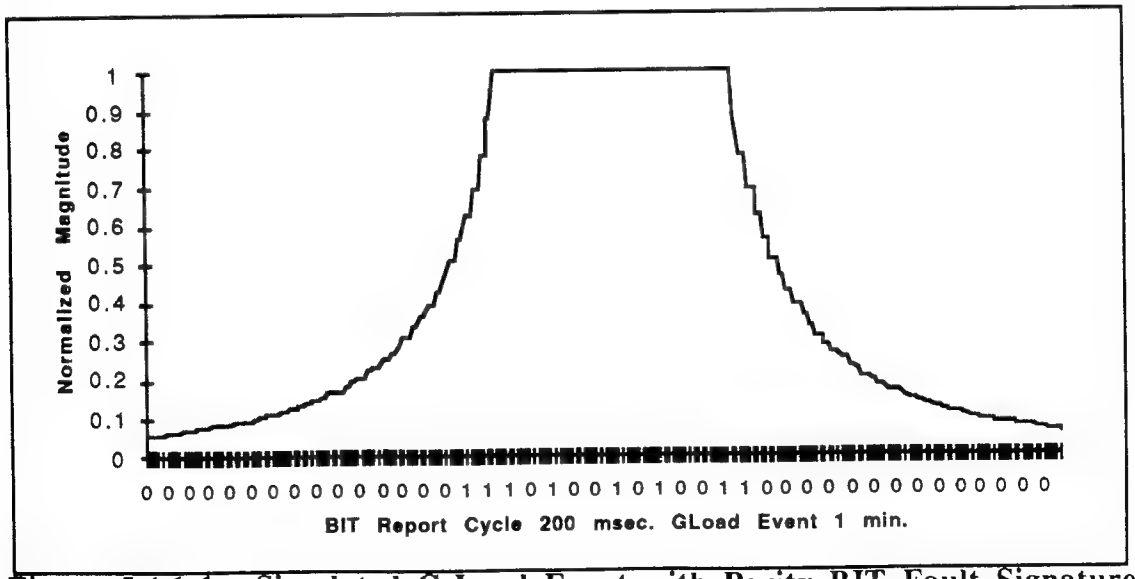


Figure 5.4.1-1. Simulated G-Load Event with Parity BIT Fault Signature

5.4.1.1 Network Model Overview

The Backpropagation network model is one of the most mature, well-known neural networks. It is a supervised network model, which means it must know the desired output for every input pattern. It is usually a three-layer network, as shown in Figure 5.4.1.1-1.

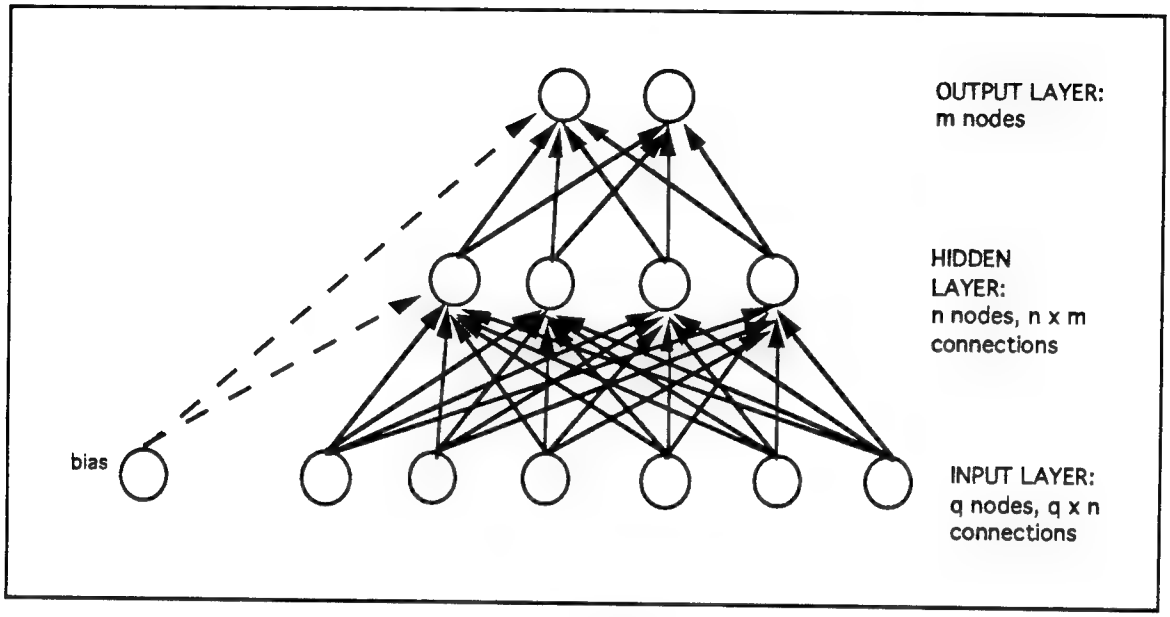


Figure 5.4.1.1-1. A Backpropagation Network Architecture

The number of nodes in the input layer is determined by the number of inputs to the network. The number of output nodes is typically defined by the number of classes that the network must learn to identify. The number of hidden nodes is determined empirically or heuristically, usually by trying several different numbers and testing to determine which is best. Briefly, the operation of a basic Backpropagation network is as follows:

There are two modes of operation, training and testing. In training mode, random real-valued weights are assigned to each connection in the network. An input data pattern is presented to the network and passed through each layer to the output layer, where the network's output is calculated. The error between the actual output and the desired output is determined at each output node. Using a gradient descent algorithm (the delta rule), the error is back propagated through the network, adjusting weights. This process is repeated many times (epochs) for all of the training patterns in the training data set. At the end of the training phase, the weights are saved to be used for testing. In testing mode, the weights are not changed. The network is run in a single pass through each testing pattern. The pattern is presented to the network and passed through each layer to the output layer where the output at each node is calculated. The values at the output nodes constitute a classification, with the maximum value corresponding to the best estimate of identification. Typically the output classes are defined using a 1 of n method, where each node represents one class and is expected to take on a value of 1 when that class has been identified. When there are clear distinctions between classes, one node will have a value significantly higher (and closer to 1) than the other nodes. That maximum value corresponds to the network classification. See the Backpropagation network tutorial in Appendix I for more information about this network.

The Backpropagation model has been applied to many problems involving pattern classification. However, it was not expected to perform well for the problem of recognizing patterns which represent streams of fault reports over intervals of time. It was selected during the down selection because it has many characteristics which made it a suitable choice, such as maturity, stability, reliability, and reasonable software size and throughput requirements. It is the most well-supported model of the NeuralWare, Inc. network tool set. It was also interesting to determine how well it would perform in this situation.

5.4.1.2 Input Data

Input data (fault signature data) was generated using the BIT simulator for the G-Load/Parity approach, and the G-load curve shown in Figure 5.4.1-1. Three sets of fault signatures were generated for training, testing, and validation, for each of the three levels of simulated BIT fault reporting (source, LRU, and GFT). Each set contained 100 of each of the 4 classes. The fault signatures for the different classes were generated by varying the Failure Threshold BIT Simulator option. The validation data sets were put aside and were not used during network development, training, or testing.

The format of the signature files can be seen in the examples in Appendix G. After they were generated, they were post-processed into a form suitable for input to the NeuralWorks Backpropagation network using a utility which was written for that purpose. Some of the network input files were made with just the BIT fault reports as input. Other input files were constructed to contain both BIT fault reports and corresponding G-load curve values to simulate enhanced environmental information.

At the source level of BIT fault reporting, the G-Load/Parity fault signatures contained 300 simulated parity BIT status reports (G-load event duration 60000 msec / BIT reporting cycle 200 msec). The number of network inputs was therefore 300 for BIT status reports only, and 600 for BIT reports with corresponding G-load data value.

At the LRU level of BIT fault reporting, the signatures contained 60 BIT status reports (G-load event duration 60000 msec / BIT reporting cycle 1000 msec). The number of network inputs was therefore 60 for BIT status reports only, and 120 for BIT reports with corresponding G-load data value.

At the GFT level of BIT fault reporting, the signatures contained BIT status reports from all of the simulated BIT devices. There were 12 groups of BIT status reports (G-load event duration 60000 msec / BIT reporting cycle 5000 msec) with 5 reports in each group. The number of network inputs was therefore 60 for BIT status reports only, and 120 for BIT reports with corresponding G-load data value.

Noise testing data was also generated using the BIT simulator's Mean and Standard Deviation options. The Mean and Standard Deviation parameters were applied to the original G-load curve file data during BIT simulator execution, resulting in a perturbed G-load curve. This was done to simulate data which would be sufficiently different from the original data that the network's robustness and generalization capabilities could be tested. Several data sets containing varying degrees of noise were generated, as shown below.

Mean Value	Standard Deviation Values
.1	.1, .5, .9
.5	.1, .5, .9
.9	.1, .5, .9

5.4.1.3 Network Design Optimization, Training and Testing

Networks needed to be built for all different numbers of inputs. For this approach, there were 2 networks built at each BIT reporting level, one for BIT status reports only, and the other for BIT reports with enhanced environmental information.

Building a Backpropagation network using the NeuralWorks Professional II/PLUS tool is very automated. Using a graphical interface, the architecture of the network is defined, input file names are identified, and network training and testing can be automatically performed.

Initial experiments were performed to determine the optimal network design. These experiments focused on the number of hidden nodes, but also varied network parameters such as learning rate and momentum. Network learning rules were also varied, using some of the set provided by NeuralWorks.

The best design was determined by building a network, setting parameters, and using the NeuralWorks' "Savebest" option to automatically perform iterative training and testing while saving the "best network so far". Savebest operates as follows: the training portion consists of randomly presenting patterns from the training data set and performing the backpropagation training as described above. Once every preset number of pattern presentations, a test pass is performed. The test pass consists of presenting each pattern in the testing data set to the network once. The overall percent correct classifications is calculated, and if the current network configuration outperformed all of the others, it is saved.

The design optimization process resulted in 6 networks. Their architecture definitions are given below. The NeuralWorks default settings for learning rate, learning rule, and momentum were used unless otherwise noted.

Table 5.4.1.3-1. G-Load/Parity Backpropagation Networks

Network	Architecture (Input-Hidden-Output)	Non-Default Setting
Source Level, fault reports only	300-20-5	None
Source Level, fault reports + environment data	600-25-5	None
LRU Level, fault reports only	60-17-5	Momentum=.2
LRU Level, fault reports + environmental data	120-15-5	Learn Rate=.5 Momentum=.8
GFT Level, fault reports only	60-25-5	None
GFT Level, fault reports + environmental data	120-25-5	None

5.4.1.4 Network Validation and Noise Testing

Once the network design was finalized, the validation was performed. Validation consisted of running the validation data set through the network once. The raw network results (written to a file

by NeuralWorks) were then processed to extract certain measures of network performance using a utility written for that purpose. The measures of performance are discussed in Section 6, Neural Network Results.

Noise testing was also performed in the same way. Each noise test file was run through the network once, and the raw results were processed to extract the overall percent correct classifications. See Section 6 for results.

5.4.2 Approach 2: Backpropagation/Vibration/Parity

Approach 2 applied the Backpropagation neural network model to the Parity BIT technique, affected by a Vibration fault report cause. Figure 5.4.2-1 shows a visualization of the simulated vibration event, overlaid with an example of a fault report signature for this approach. The duration of the vibration event was defined to be 2 seconds (2000 msec) and a corresponding vibration environmental curve file was generated (see Appendix F). The parity BIT simulator for this approach is described in section 4.3.3.4.1. It was designed to produce a BIT report every 20 msec.

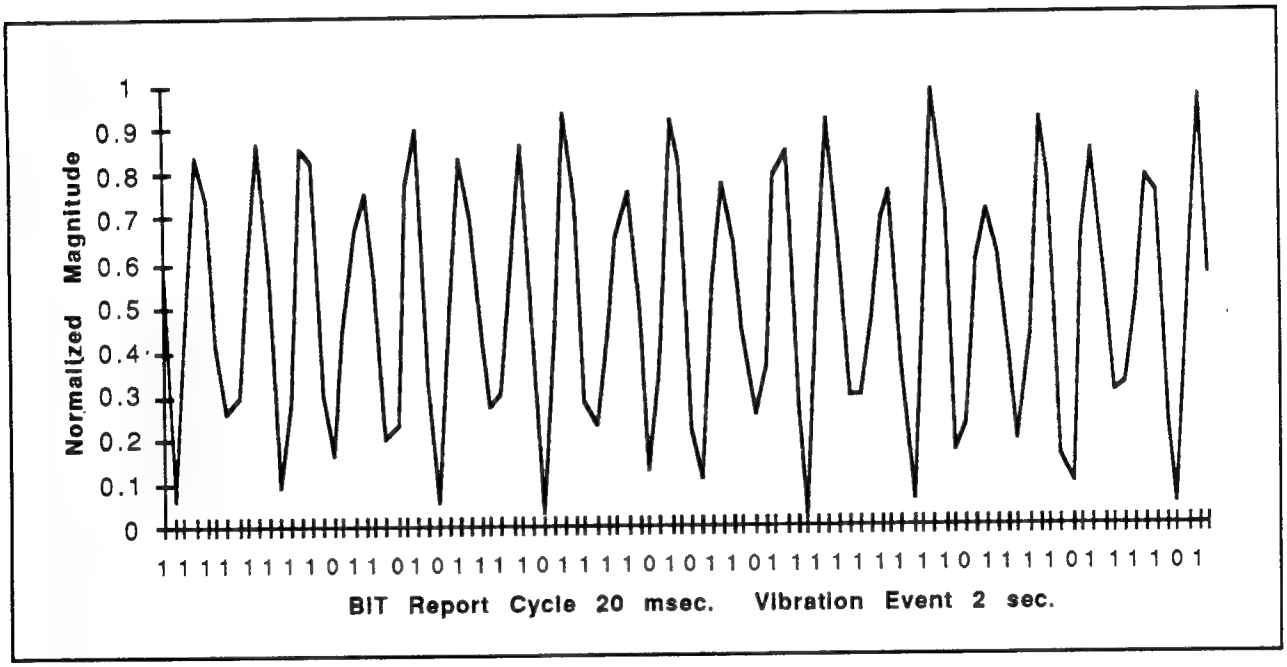


Figure 5.4.2-1. Simulated Vibration Event with Parity BIT Fault Signature

5.4.2.1 Network Model Overview

The Backpropagation network model was used for this approach. It is described in Section 5.4.1.1.

5.4.2.2 Input Data

Input data (fault signature data) was generated using the BIT simulator for the Vibration/Parity approach, and the Vibration curve shown in Figure 5.4.2-1. Three sets of fault signatures were

generated for training, testing, and validation, for each of the three levels of simulated BIT fault reporting (source, LRU, and GFT). Each set contained 100 of each of the 4 classes. The fault signatures for the different classes were generated by varying the Failure Threshold BIT Simulator option. The validation data sets were put aside and were not used during network development, training, or testing.

The format of the signature files can be seen in the examples in Appendix G. After they were generated, they were post-processed into a form suitable for input to the NeuralWorks Backpropagation network using a utility which was written for that purpose. Some of the network input files were made with just the BIT fault reports as input. Other input files were constructed to contain both BIT fault reports and corresponding vibration curve values to simulate enhanced environmental information.

At the source level of BIT fault reporting, the Vibration/Parity fault signatures contained 100 simulated parity BIT status reports (vibration event duration 2000 msec / BIT reporting cycle 20 msec). The number of network inputs was therefore 100 for BIT status reports only, and 200 for BIT reports with corresponding vibration data value.

At the LRU level of BIT fault reporting, the signatures contained 2 BIT status reports (vibration event duration 2000 msec / BIT reporting cycle 1000 msec). The number of network inputs was therefore 2 for BIT status reports only, and 4 for BIT reports with corresponding vibration data value. (This was not enough input data to allow the network to make any meaningful classifications, as discussed below.)

At the GFT level of BIT fault reporting, the signatures contained BIT status reports from all of the simulated BIT devices. There was 1 group of BIT status reports (vibration event duration 2000 msec / BIT reporting cycle 5000 msec) with 5 reports in each group. The number of network inputs was therefore 5 for BIT status reports only, and 10 for BIT reports with corresponding vibration data value. (This was not enough input data to allow the network to make any meaningful classifications, as discussed below.)

Noise testing data was also generated using the BIT simulator's Mean and Standard Deviation options. The Mean and Standard Deviation parameters were applied to the original vibration curve file data during BIT simulator execution, resulting in a perturbed vibration curve. This was done to simulate data which would be sufficiently different from the original data that the network's robustness and generalization capabilities could be tested. Several data sets containing varying degrees of noise were generated, as shown below.

Mean Value	Standard Deviation Values
.01	.01, .1
.03	.01, .1
.05	.01, .1
.1	.1, .5, .9
.5	.1, .5, .9

5.4.2.3 Network Design, Optimization, Training and Testing

As for the previous approach, networks needed to be built for all different numbers of inputs. For this approach, there were 2 networks built at each BIT reporting level, one for BIT status reports only, and the other for BIT reports with enhanced environmental information.

Initial experiments were performed to determine the optimal network design, using the NeuralWorks Backpropagation network builder, and the Savebest option for training and testing. These experiments focused on the number of hidden nodes, but also varied network parameters such as learning rate and momentum. Network learning rules were also varied, using some of the set provided by NeuralWorks

The design optimization process resulted in 2 networks for the source level of BIT fault reporting. Their architecture definitions are given below. The NeuralWorks default settings for learning rate, learning rule, and momentum were used unless otherwise noted.

Table 5.4.2.3-1. Vibration/Parity Backpropagation Networks

Network	Architecture (Input-Hidden-Output)	Non-Default Setting
Source Level, fault reports only	100-5-5	None
Source Level, fault reports + environment data	200-5-5	None

The amount of information in the higher level fault signatures was not enough to be used as valid network input. Experiments were performed at the LRU level to determine the minimum number of inputs which would produce meaningful network classifications, and the required BIT status reporting cycle rate. At a BIT status report cycle rate of 50 msec, a signature with 40 fault reports was sufficient for good classifications.

5.4.2.4 Network Validation and Noise Testing

Validation for this approach was only done with the source level networks. As in the previous approach, validation consisted of running the validation data set through the network once. The raw network results (written to a file by NeuralWorks) were then processed to extract certain measures of network performance using a utility written for that purpose. The measures of performance are discussed in Section 6, Neural Network Results.

Noise testing was also performed in the same way. Each noise test file was run through the network once, and the raw results were processed to extract the overall percent correct classifications. See Section 6 for results.

5.4.3 Approach 3: SPR/Temperature/Activity Detector

Approach 3 applied the SPR neural network model to the Activity Detector BIT technique, affected by a Temperature fault report cause. Figure 5.4.3-1 shows a visualization of the simulated temperature event, overlaid with an example of a fault report signature for this approach. The duration of the temperature event was defined to be 20 minutes (1200000 msec) and a corresponding temperature environmental curve file was generated (see Appendix F). The activity detector BIT simulator for this approach is described in section 4.3.3.2.1. It was designed to produce a BIT report every 20 msec.

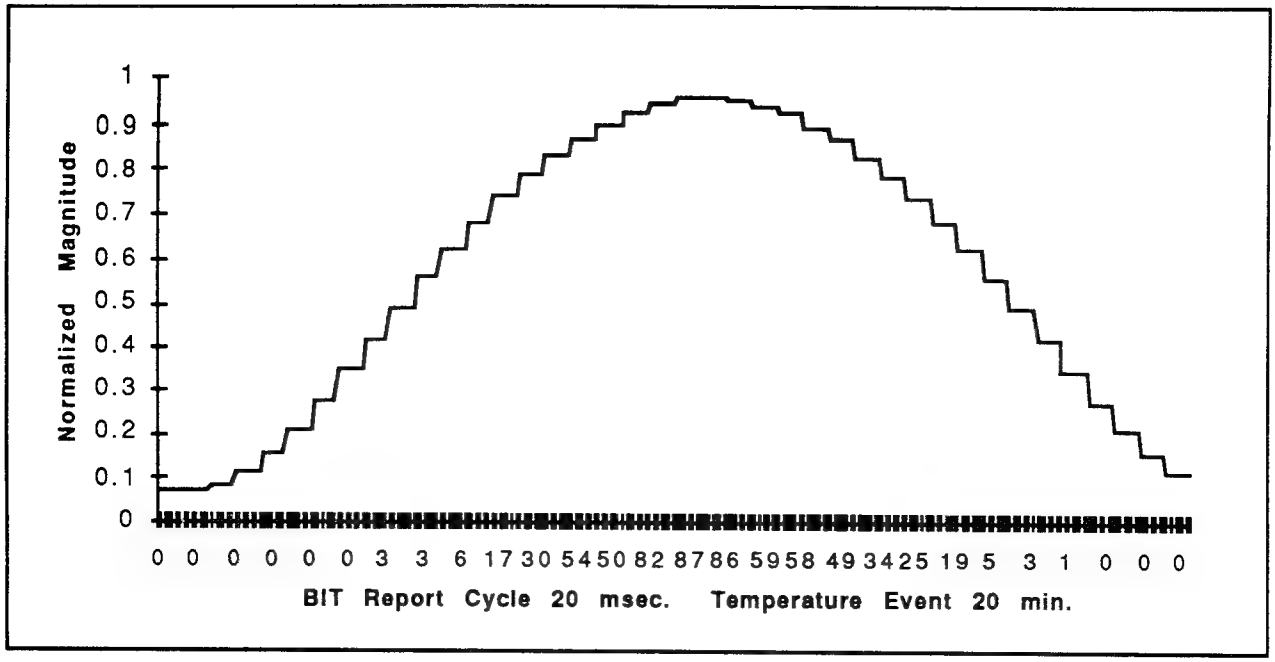


Figure 5.4.3-1. Simulated Temperature Event with Activity Detector BIT Fault Signature

5.4.3.1 Network Model Overview

The SPR network is less well-known than Backpropagation. It is a network which has been used to recognize sequences of events and is therefore suited to time-varying problems such as BIT false alarm filtering. It is provided on the NeuralWorks network builder menu. Its architecture, as implemented in NeuralWorks, is shown in Figure 5.4.3.1-1. There is a normalized input layer, connected to a set of "detection chains" which appear as rows in the figure. Each row represents the path to an output node, which in turn represents a data class. The path to the output passes through time slices, which are the columns in the figure. The input is presented in normalized form to the network, and is passed through each time slice in what is called an "avalanche" of activity through the detection chain. The output node with the highest value (closest to 1) represents the network's classification of the input pattern. This is also a supervised network, so that desired outputs must be presented to the network along with the corresponding input pattern. Learning takes place using a Kohonen-based learning rule. See the SPR network tutorial in Appendix I for more information about this network.

The SPR model has been applied to certain classification problems such as repetitive signal recognition which involve viewing a data pattern as a series of temporal events. Initially, it was an alternate to the Backpropagation Through Time (BPTT) network, which was the primary candidate for this approach. BPTT is not provided by NeuralWorks, but NeuralWorks provides the capability to construct and run external networks within their development environment. However, the underlying network execution mechanisms in the tool did not lend themselves to building BPTT networks within time and budget constraints. Therefore, the alternate SPR was used.

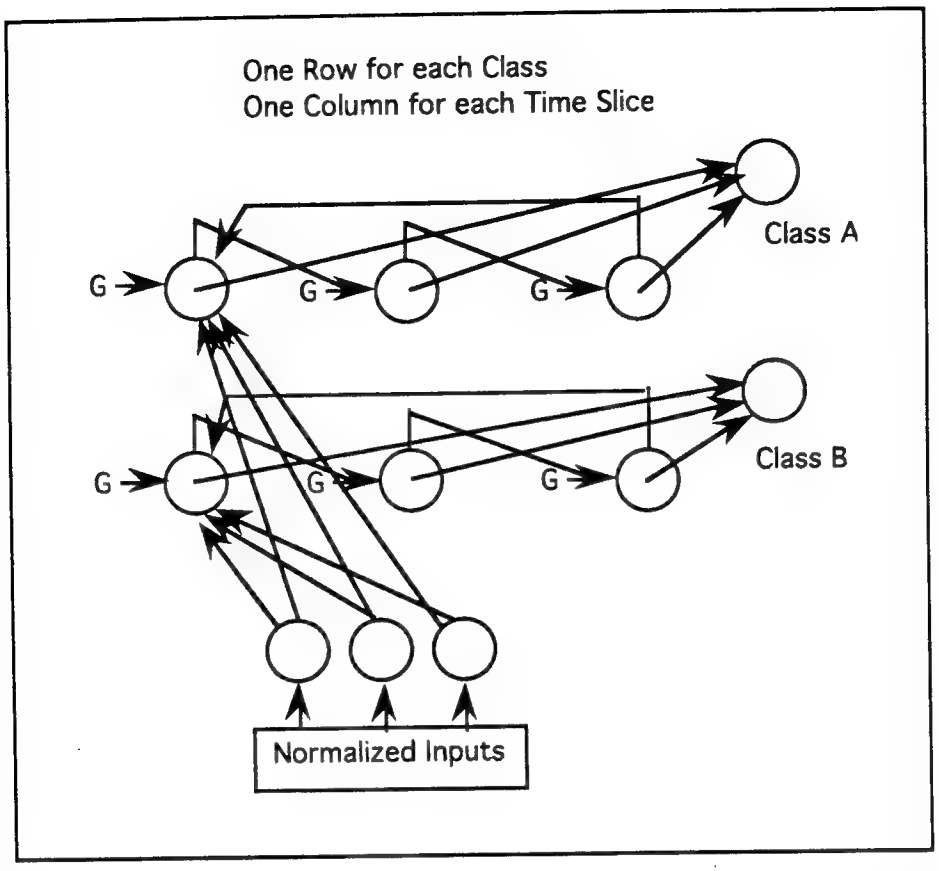


Figure 5.4.3.1-1. The NeuralWorks SPR Network Architecture

5.4.3.2 Input Data

Input data (fault signature data) was generated using the BIT simulator for the Temperature/Activity Detector approach, and the temperature curve shown in Figure 5.4.3-1. Three sets of fault signatures were generated for training, testing, and validation, for each of the three levels of simulated BIT fault reporting (source, LRU, and GFT). Each set contained 100 of each of the 4 classes. The fault signatures for the different classes were generated by varying the Failure Threshold BIT Simulator option. The validation data sets were put aside and were not used during network development, training, or testing.

The format of the signature files can be seen in the examples in Appendix G. After they were generated, they were post-processed into a form suitable for input to the NeuralWorks SPR network using a utility which was written for that purpose. Some of the network input files were made with just the BIT fault reports as input. Other input files were constructed to contain both BIT fault reports and corresponding temperature curve values to simulate enhanced environmental information.

Initially, at the source level of BIT fault reporting, the Temperature/Activity Detector fault signatures contained 60000 simulated parity BIT status reports (temperature event duration 1200000 msec / BIT reporting cycle 20 msec). The number of network inputs (60000) was prohibitively large. The BIT simulator for this approach was modified to collect running sums of

the number of simulated BIT device faults for 200 msec periods. The sum value was then written to the fault signature file. This resulted in a source level fault signature which contained 300 BIT status reports (1200000 / 20 / 200). The number of network inputs was therefore 300 for BIT status reports only, and 600 for BIT reports with corresponding temperature data value. The same process was performed for the LRU and GFT levels.

Noise testing data was also generated using the BIT simulator's Standard Deviation option. The standard deviation was applied to the original temperature curve file data during BIT simulator execution, resulting in a perturbed temperature curve. This was done to simulate data which would be sufficiently different from the original data that the network's robustness and generalization capabilities could be tested. Several data sets containing varying degrees of noise were generated, with standard deviation set to .01, .05, .10, .20, .30, and .50.

5.4.3.3 Network Design Optimization, Training and Testing

Networks needed to be built for all different numbers of inputs. For this approach, there were 2 networks built at each BIT reporting level, one for BIT status reports only, and the other for BIT reports with enhanced environmental information.

Building an SPR network using the NeuralWorks Professional II/PLUS tool is also automated to some degree. Using the graphical interface, the architecture of the network can be defined and input file names can be identified. The Savebest option is not available for SPR nets. However, NeuralWare provides an off-line utility called Automate that allows a user to script training and testing scenarios which mimic the Savebest process. Automate was used to automate SPR network training and testing.

Initial experiments were performed to determine the optimal network design. These experiments focused on determining the optimal number of time slices, whether or not to allow automatic normalization of the network inputs, and how many times to run an input pattern test. SPR network training, testing, and validation is significantly different from all of the other networks in the method of input pattern presentation. Each pattern must be presented individually, and the pattern test must be cycled several times while the network settles on the result. Patterns to be presented were placed in individual input files. Since the Savebest option was not available, Automate was used to load a network, present one input file, cycle the testing, and continue to the next input file until all patterns had been presented. For every input file, NeuralWorks produced a raw results file. These files were processed by a custom-built utility, to determine which were the best-performing networks.

The design optimization process resulted in 6 networks, each with 300 inputs, using 3 time slices (100 BIT reports each) and 4 classes. The automatic polar normalization was always beneficial so it was used for all networks. All other settings were defaults.

5.4.3.4 Network Validation and Noise Testing

Validation for this approach was done as described above for training/testing. Individual patterns were presented as input files, raw result files were generated by NeuralWorks and then processed to extract certain measures of network performance using a utility written for that purpose. The measures of performance are discussed in Section 6, Neural Network Results.

Noise testing was also performed in the same way. Each noise test pattern file was run through the network test cycle, and the raw results were processed to extract the overall percent correct classifications. See Section 6 for results.

5.4.4 Approach 4: REINFORCE/Temperature/Viterbi Decoder

Approach 4 applied the REINFORCE neural network model to the Viterbi Decoder BIT technique, affected by a Temperature fault report cause. Figure 5.4.4-1 shows a visualization of the simulated temperature event, overlaid with an example of a fault report signature for this approach. The duration of the temperature event was defined to be 20 minutes (1200000 msec) and a corresponding temperature environmental curve file was generated (see Appendix F). The Viterbi decoder BIT simulator for this approach is described in section 4.3.3.1.1. It was designed to produce a BIT report every 20 msec.

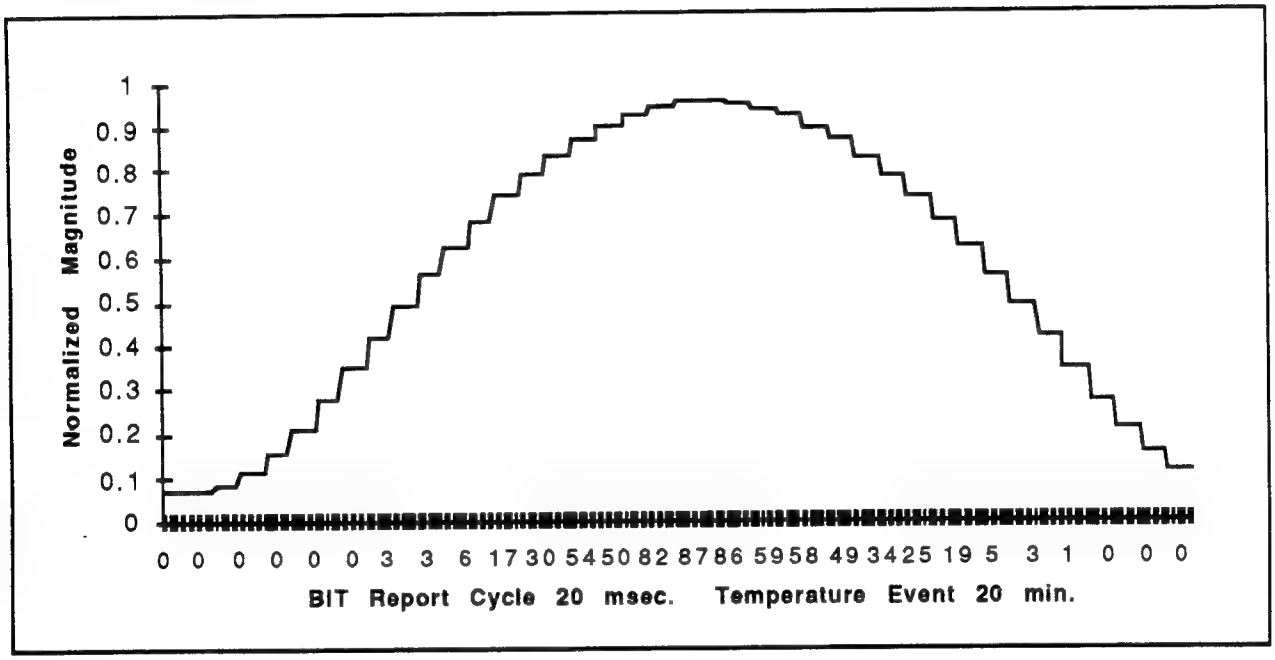


Figure 5.4.4-1. Simulated Temperature Event with Viterbi Decoder BIT Fault Signature

5.4.4.1 Network Model Overview

The REINFORCE network is more experimental than the other two. It is not provided within the NeuralWorks tool framework. It is a more adaptive model which fits into the reinforcement learning category of networks: it does not require the desired output to be presented with the input pattern. Theoretically it could learn "on-line", without previous supervisory training. Its general architecture is shown in Figure 5.4.4.1-1.

Basic operation of a REINFORCE network is as follows: In training mode, an input pattern is presented to the network (it comes in from the environment, or the network's external world). As the input pattern is presented to each network input node, the node outputs a value which represents its action in response to the input. These actions, or activations, pass through the network from input side to output side. After all the nodes at the output side have output their

action, (the resulting classification), the environment calculates an evaluation (reinforcement) according to the particular network input pattern. Each node then responds to the reinforcement by changing its internal state according to some specific function of its current state, its output, its input, and the reinforcement. The precise manner in which the reinforcement signal is used depends on the learning algorithm which is applied. In the typical case, the reinforcement signal is broadcast to all nodes. In testing mode, the reinforcement from the environment is removed. The network receives input, passes the input through to the outputs, and the result is the network classification. The number of outputs determines how the result is formulated. See the REINFORCE network tutorial in Appendix I for more information about this network.

The REINFORCE model had not been applied to many real-world classification or other problems. It was selected during the down selection because it had the potential to perform well with temporal data, it would not present unreasonable storage or timing requirements, and it appeared to be more well-defined and documented than other theoretical techniques which were under consideration.

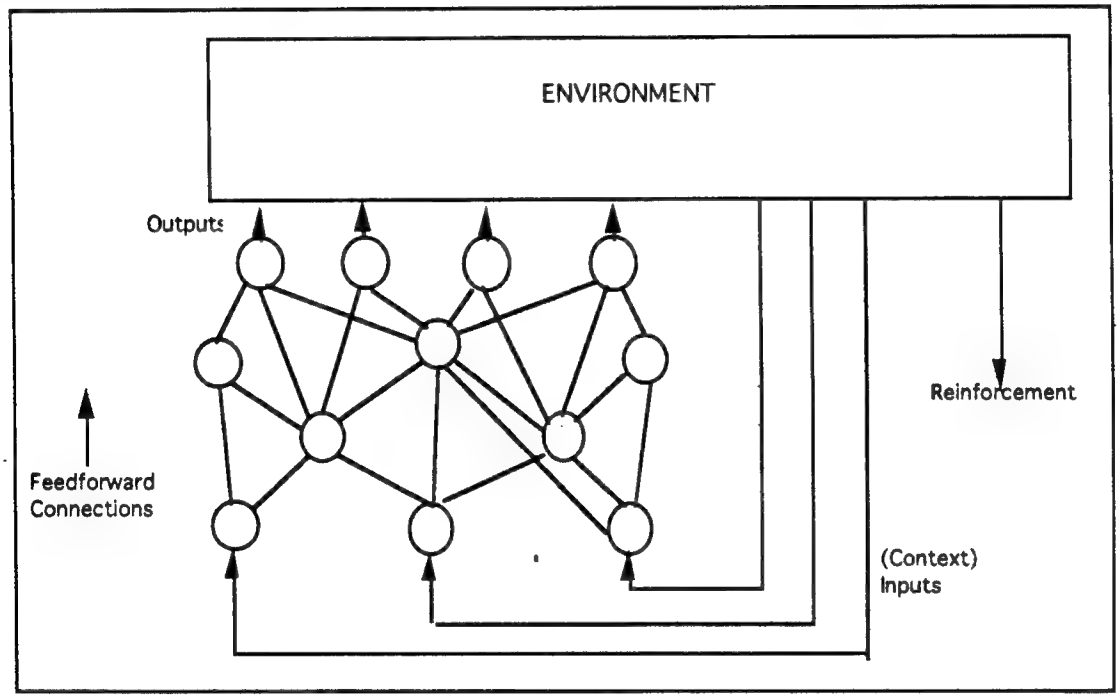


Figure 5.4.4.1-1. A General REINFORCE Network Architecture

5.4.4.2 Input Data

Input data (fault signature data) was generated using the BIT simulator for the Temperature/Viterbi Decoder approach, and the temperature curve shown in Figure 5.4.4-1. Three sets of fault signatures were generated for training, testing, and validation, for each of the three levels of simulated BIT fault reporting (source, LRU, and GFT). Each set contained 100 of each of the 4 classes. The fault signatures for the different classes were generated by varying the Failure Threshold BIT Simulator option. The validation data sets were put aside and were not used during network development, training, or testing.

The format of the signature files can be seen in the examples in Appendix G. After they were generated, they were post-processed into a form suitable for input to the REINFORCE network using a utility which was written for that purpose. Since the environmental data was used in calculation of the reinforcement signal, it was not added to the fault signature files. Initially, there were the same network input size problems as the SPR network. The Viterbi Decoder BIT simulator generated two-bit fault reports for this approach, since the Viterbi Decoder can output 3 values (0=no fault, 1=error detected and corrected, 2=uncorrectable error). At the source level of BIT fault reporting, the Temperature/Viterbi Decoder fault signatures contained 1200000 simulated Viterbi Decoder BIT status reports (temperature event duration 1200000 msec / BIT reporting cycle 20 msec * 2 fault bits). The number of network inputs (1200000) was prohibitively large. The BIT simulator for this approach was modified to collect running sums of the number of simulated BIT device faults for 400 msec periods. The sum value was then written to the fault signature file. This resulted in a source level fault signature which contained 300 BIT status reports (1200000 / 20 * 2 / 400). The number of network inputs was therefore 300. The same process was performed for the LRU and GFT levels.

Noise testing data was also generated using the BIT simulator's Standard Deviation option. The standard deviation was applied to the original temperature curve file data during BIT simulator execution, resulting in a perturbed temperature curve. This was done to simulate data which would be sufficiently different from the original data that the network's robustness and generalization capabilities could be tested. Several data sets containing varying degrees of noise were generated, with standard deviation set to .01, .05, .10, .20, .30, and .50.

5.4.4.3 Network Design Optimization, Training and Testing

Since the REINFORCE network was not provided by NeuralWorks, NeuralWare's User Defined Neuro-Dynamics (UDND) product was used to incorporate the REINFORCE network's mathematical and other computational functions into the structure of the NeuralWorks development environment. A function was defined for four critical network node calculations: sum, output, transfer function, and learning.

Once the new network functions were incorporated into the NeuralWorks tool framework, the tool could then be used to build, train, and test the networks. As for the previous approaches, networks needed to be built for all different numbers of inputs. For this approach, there was 1 network built at each BIT reporting level. Initial experiments were performed to determine the optimal network design, using the Savebest option for training and testing. These experiments focused on the calculation of the reinforcement signal, the layering of the network (whether or not to use hidden nodes), the connectivity of the network, the number of classes and class representation, and learning rate. The design optimization process resulted in 3 networks, one at each level of BIT fault reporting. Each had 300 inputs, 0 hidden nodes, and 1 output node, with a learning rate of .01. The network classification at the one output node was designed as a range of values from 0 to 1, with empirically determined thresholds to mark the delimitation between the four classes. A reinforcement function was defined that utilized the temperature data along with the number of faults in the input pattern.

5.4.4.4 Network Validation and Noise Testing

Validation consisted of running the validation data set through the network once. The raw network results (written to a file by NeuralWorks) were then processed to extract certain measures of network performance using a utility written for that purpose. The measures of performance are discussed in Section 6, Neural Network Results. Noise testing was also performed in the same

way. Each noise test file was run through the network once, and the raw results were processed to extract the overall percent correct classifications. See Section 6 for results.

6. NEURAL NETWORK RESULTS

This section presents the results of testing the performance of each of the neural networks using validation data (non-noisy) and using noisy data. The validation data sets were generated using the NNFAF BIT simulator. Uniqueness from any previously seen training or testing data was ensured by using the BIT simulator option to seed the random number generator with a different seed for the validation data. Data was generated for the three simulated levels of BIT fault reporting (source, LRU, GFT). Data was generated with and without additional environmental information.

The noisy data was generated using the NNFAF BIT simulator. Different amounts of noise were applied to the data by manipulating the BIT simulator standard deviation option. The amount of noise applied to the data sets differed from approach to approach. Generally, the amounts of noise can be categorized as low or moderate. They are described within the individual approach sections which follow.

The noise test results presented in each of the following sections indicate the amount of noise perturbation applied to the data. The amount of perturbation was calculated using the L2 norm:

$$\sqrt[2]{\sum(f-f')^2 \div \sum f^2}$$

where f is the original data value and f' is the perturbed value.

Three measures of goodness were defined to describe the performance of each network. They are summarized in Table 6-1. Parameters P1 and P2 distinguish between fault reports which do or do not require a repair action. Those which require a repair action encompass both Intermittent and Hard Failures. Those that do not require a repair action encompass both No Faults as well as False Alarms. In the False Alarm case, a fault report is generated but it should not result in any repair action. The two parameters reflect network false alarm filtering capability as well as the ability to recognize and report a failure. Parameter P3 represents the overall network performance, i.e., how well it was able to correctly classify each of the fault report signatures in the validation data set.

Table 6-1. Neural Network Performance Measures of Goodness

Goodness Parameter	Description	How Calculated
P1	Percent Correctly Classified as Requiring a Repair Action	(Total Correct Intermittent + Total Correct Hard Fault) / Total Intermittent + Hard Fault in Test Data Set
P2	Percent Correctly Classified as Not Requiring any Repair Action	(Total Correct No Fault + Total Correct False Alarm) / Total No Fault + False Alarm in Test Data Set
P3	Overall Percent Correct	(Total Correct No Fault + Total Correct False Alarm + Total Correct Intermittent + Total Correct Hard Fault) / Total in Test Data Set

In the sections that follow, the network results are presented using classification matrices (also known as confusion matrices). An example is shown in Figure 6-1. The matrix columns contain the actual network classifications, and the rows contain the desired classifications. The column and row headers contain class id abbreviations (NO = No Fault, FA = False Alarm, INT = Intermittent, and HA = Hard Fault). The top-left-to-lower-right diagonal contains the correct network classifications for each class. Any entry which is not on this diagonal is a misclassification. The total number of fault report signatures in the data sets was typically 400, with 100 of each class. Therefore, the values in the matrices represent both actual quantities as well as percentages.

Desired Classification \ Actual Classification	NO	FA	INT	HA
NO	100			
FA		98	2	
INT			100	
HA				100

Figure 6-1. Example Network Classification Matrix

6.1 Approach 1 (Backpropagation/G-Load/Parity) Results

6.1.1 Validation Test Results

The validation test results for Approach 1 are shown in Figures 6.1.1-1 through 6.1.1-3. Results are presented both without and with enhanced environmental information. Figure 6.1.1-1 presents the results at the source level of BIT fault reporting, Figure 6.1.1-2 shows the results at the LRU level, and Figure 6.1.1-3 shows the results at the GFT level. All results are quite good; the LRU level is slightly better than the source level, and significantly better than the GFT level especially at distinguishing between intermittent failures and false alarms. The inclusion of environmental data in the network input was slightly beneficial at all levels of reporting.

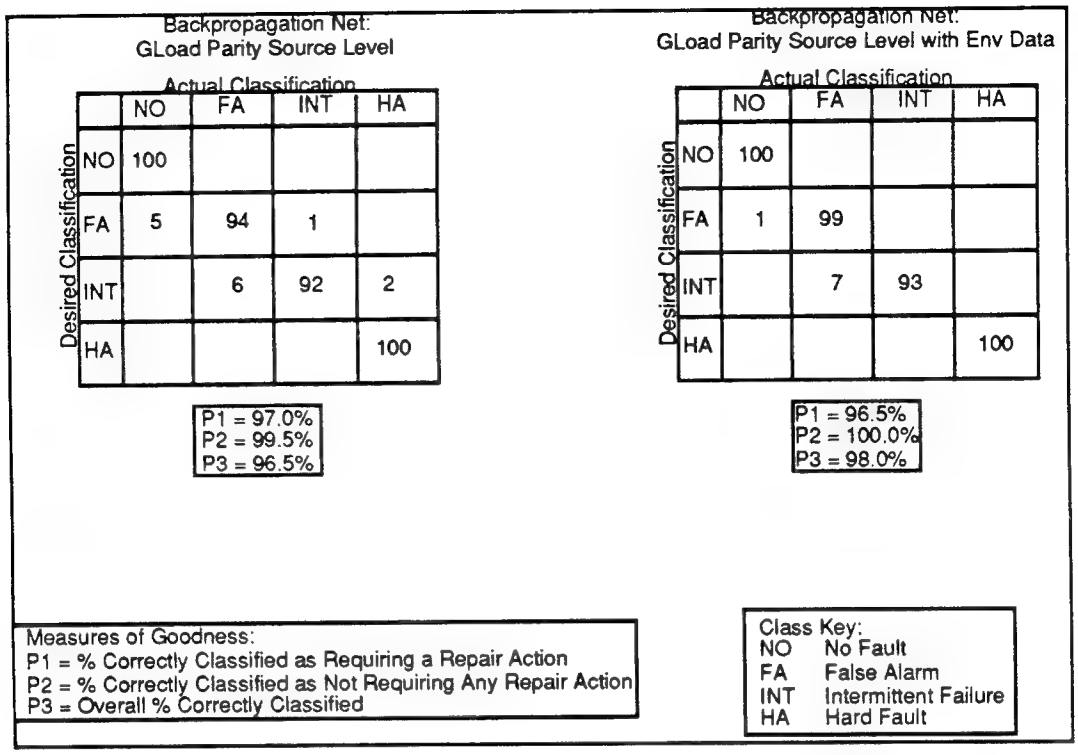


Figure 6.1.1-1. Validation Results Approach 1, Source BIT Reporting Level

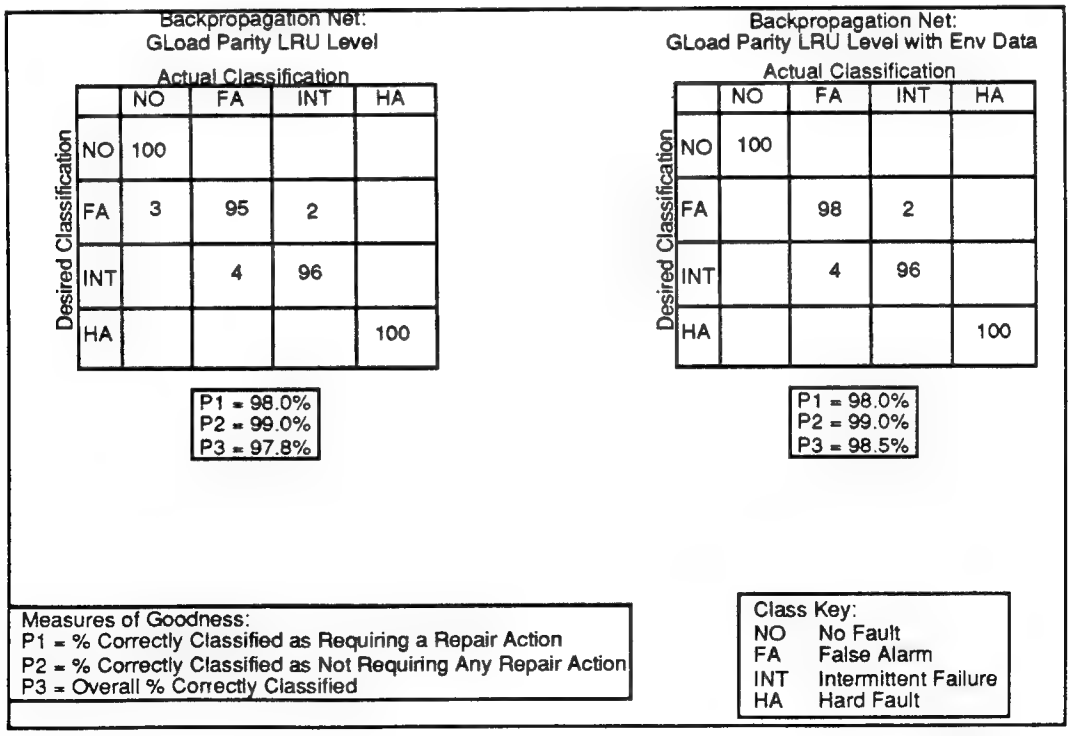


Figure 6.1.1-2. Validation Results Approach 1, LRU BIT Reporting Level

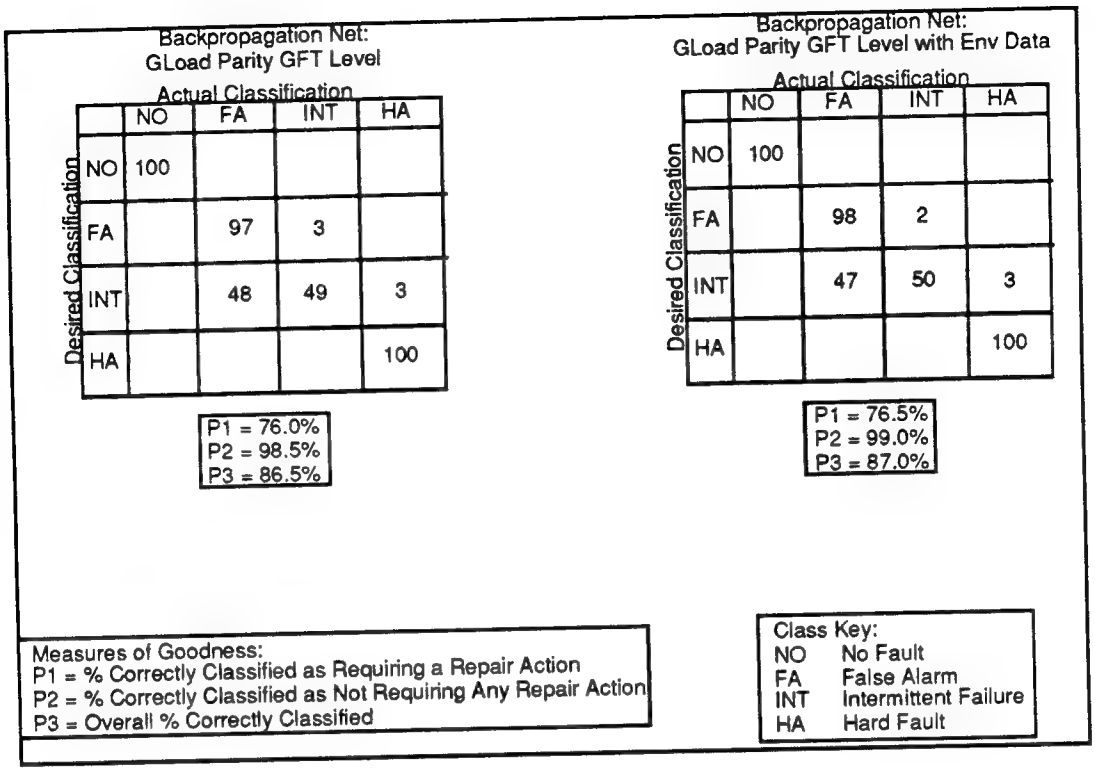


Figure 6.1.1-3. Validation Results Approach 1, GFT BIT Reporting Level

6.1.2 Noise Test Results

Figures 6.1.2-1 through 6.1.2-3 show the results of noise testing for Approach 1. The addition of noise to the test data significantly affected network performance. For this approach, large perturbations of the data were possible, most likely because the characteristics of the G-Load event are very different from the noise which was added. As was seen for all network models, the backpropagation network was always able to correctly classify No Faults and Hard Faults, so that the network performance leveled off at approximately 50% overall correct classifications, regardless of the amount of noise added. The addition of environmental data was slightly beneficial to network performance with noisy data, especially at the LRU level of BIT reporting.

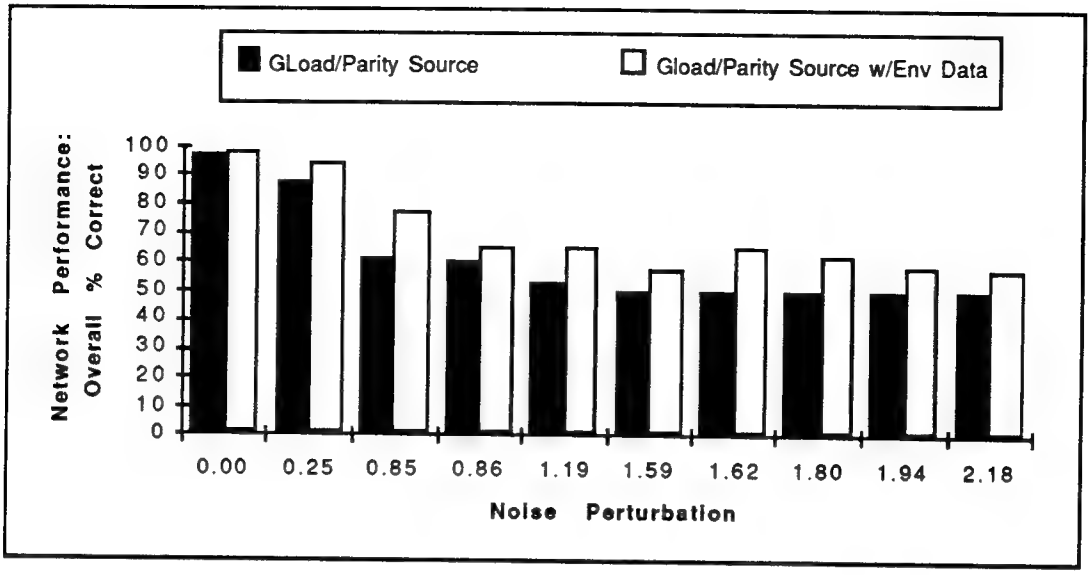


Figure 6.1.2-1. Noise Testing Approach 1, Source BIT Reporting Level

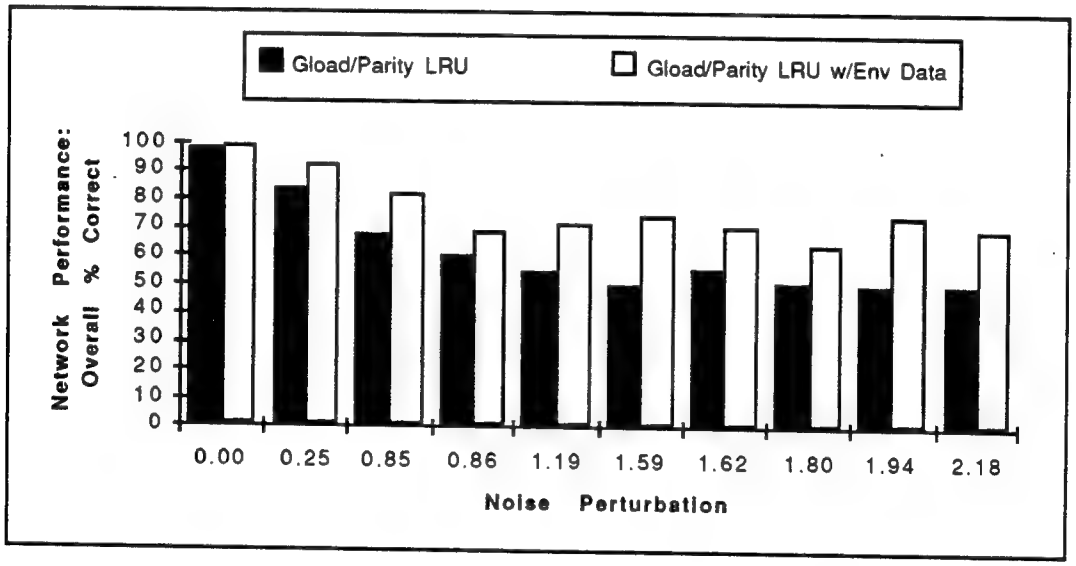


Figure 6.1.2-2. Noise Testing Approach 1, LRU BIT Reporting Level

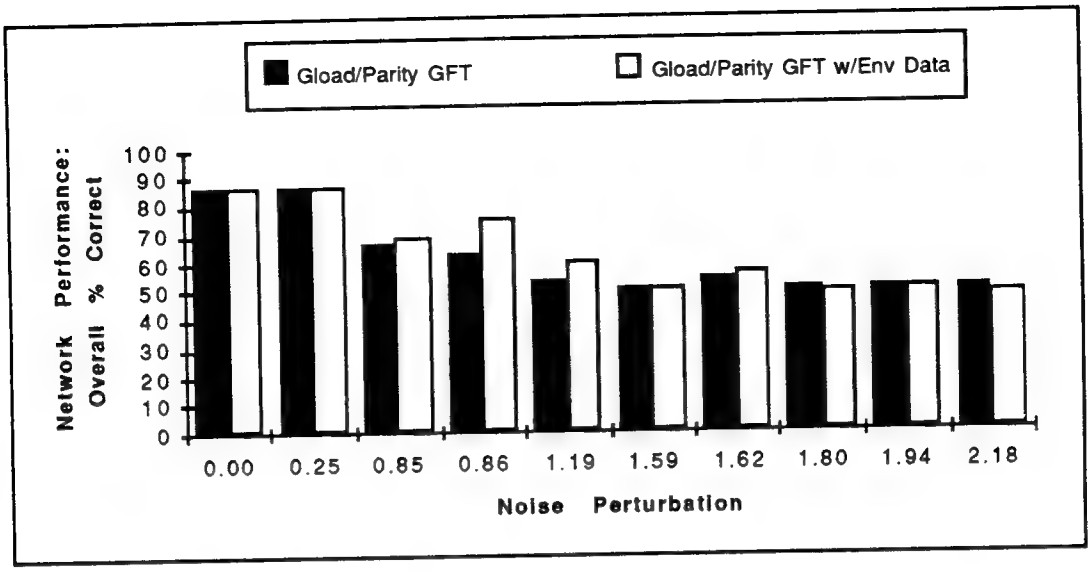


Figure 6.1.2-3. Noise Testing Approach 1, GFT BIT Reporting Level

6.1.3 Summary

The backpropagation network performed surprisingly well with this approach, as indicated in the validation result figures. This was not initially expected, because backpropagation is not usually used with temporal data. We saw differences in performance across the different levels of BIT reporting, most likely due to the amount of information in the data: the LRU level data seemed to have an optimal balance of information content, whereas the GFT level data was probably too compressed. We noted that the environmental enhancement was increasingly beneficial as the data became noisier. We concluded that the backpropagation network is a viable candidate for future study in this problem domain.

6.2 Approach 2 (Backpropagation/Vibration/Parity) Results

6.2.1 Validation Test Results

Figure 6.2.1-1 shows the results of the validation testing of Approach 2, for data without and with environmental information enhancement. Only source level results are provided, since there were no meaningful results at either the LRU or the GFT levels. The results are very good; the backpropagation network performed with 100% correct classifications on the previously unseen data. The inclusion of environmental data in the network input was slightly beneficial.

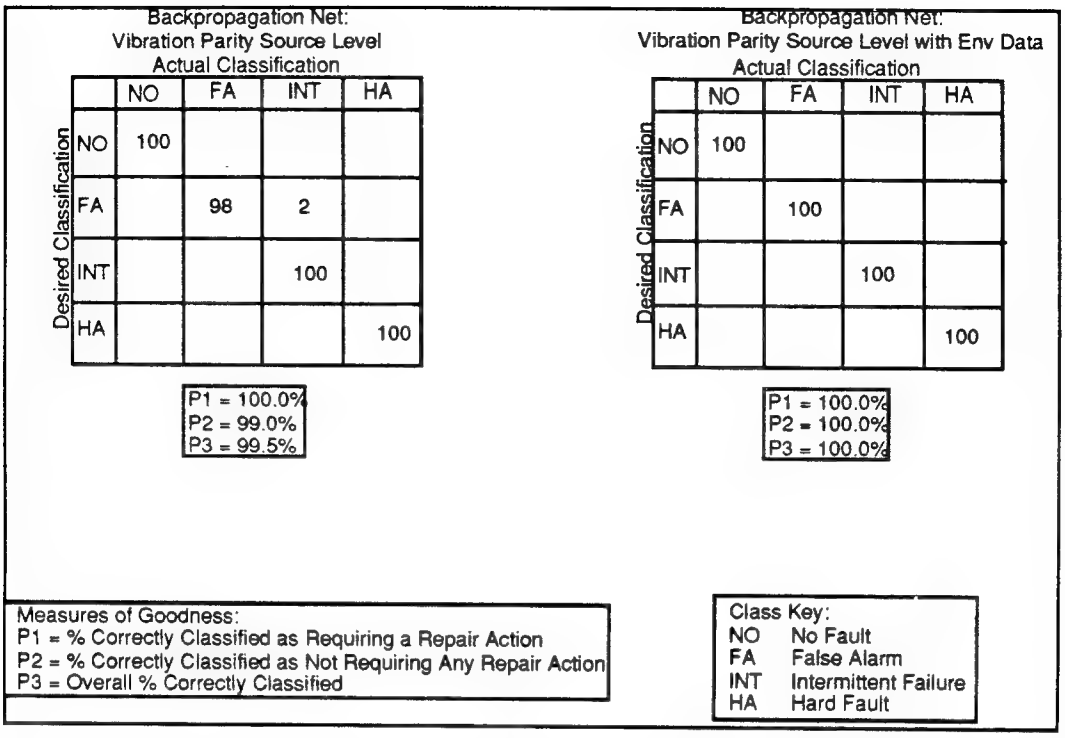


Figure 6.2.1-1. Validation Results Approach 2, Source BIT Reporting Level

6.2.2 Noise Test Results

Figure 6.2.2-1 shows the results of noise testing for Approach 2. The addition of noise to the test data significantly affected network performance. For this approach, slight variations in noise caused wide fluctuations in performance, possibly because the network was unable to distinguish between the characteristics of the noise vs. the vibration event. As was seen for all network models, the backpropagation network was always able to correctly classify No Faults and Hard Faults until the amount of noise became excessive (at 16%). The addition of environmental data was slightly beneficial to network performance with noisy data.

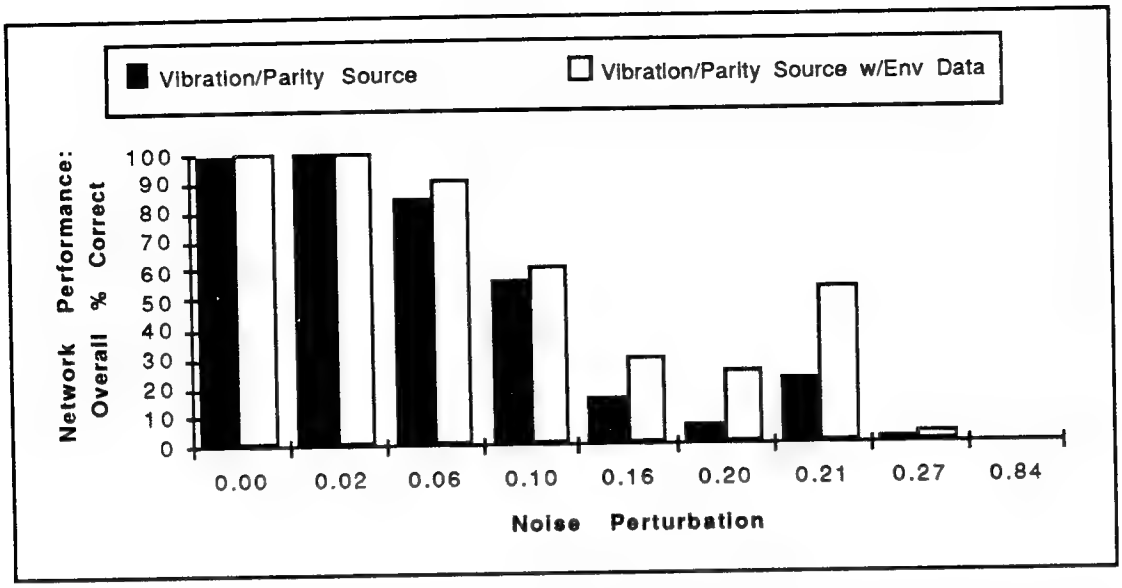


Figure 6.2.2-1. Noise Testing Approach 2, Source BIT Reporting Level

6.2.3 Summary

At the simulated source level of BIT fault reporting, results were very successful for the backpropagation network as it was applied to the parity BIT technique with a vibration fault report cause. The results were so successful that there was only a slight improvement with the data which was enhanced with information about the vibration event. Results were not meaningful at the higher levels of fault reporting because of the brief duration of the simulated vibration event. At the higher levels of reporting, the source level BIT reports were compressed by 'oring' them over the reporting period. The data was so compressed for this particular approach that at the LRU level of reporting, there were 2 fault report signatures per reporting cycle, and at the GFT level there was only one. This was not enough information for the network to be properly trained or to make meaningful classifications.

The backpropagation network requires a certain amount of unique data in order to be properly trained. We saw that, in our simulations, certain types of information could only be seen within certain intervals of time. Therefore, we conclude that is critical to status the BIT fault reports within a long enough time window such that information about transient events (such as the vibration event in this approach) will not be lost due to data compression.

6.3 Approach 3 (SPR/Temperature/Activity Detect) Results

6.3.1 Validation Test Results

The validation test results for Approach 3 are shown in Figures 6.3.1-1 through 6.3.1-3. Results are presented both without and with enhanced environmental information. Figure 6.3.1-1 presents the results at the source level of BIT fault reporting, Figure 6.3.1-2 shows the results at the LRU level, and Figure 6.3.1-3 shows the results at the GFT level. The SPR networks were not able to distinguish between No Faults and False Alarms, or between Intermittents and Hard Faults. The inclusion of environmental data was detrimental at all levels. The networks were able to separate the data into two categories which correspond to the P1 and P2 measures of goodness.

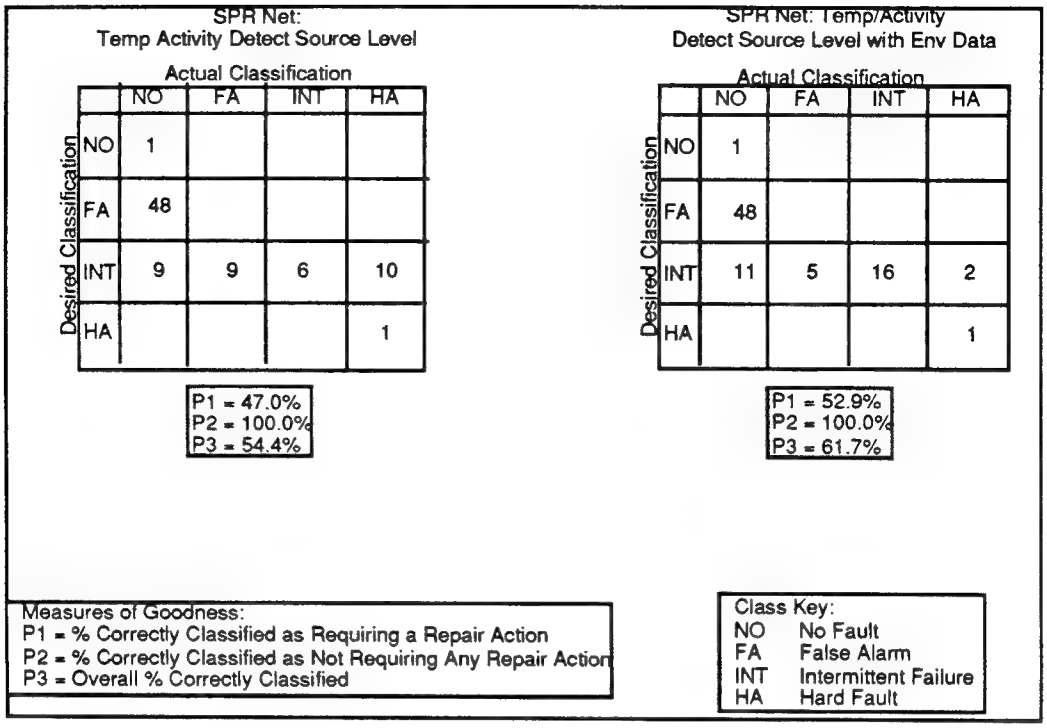


Figure 6.3.1-1. Validation Results Approach 3, Source BIT Reporting Level

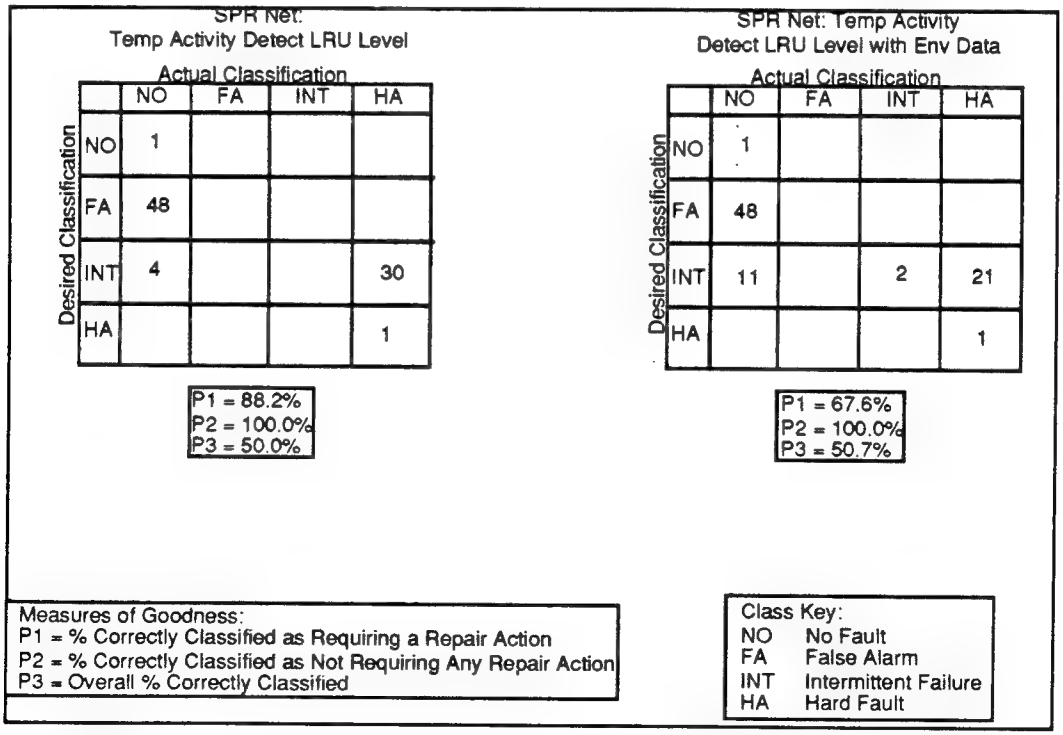


Figure 6.3.1-2. Validation Results Approach 3, LRU BIT Reporting Level

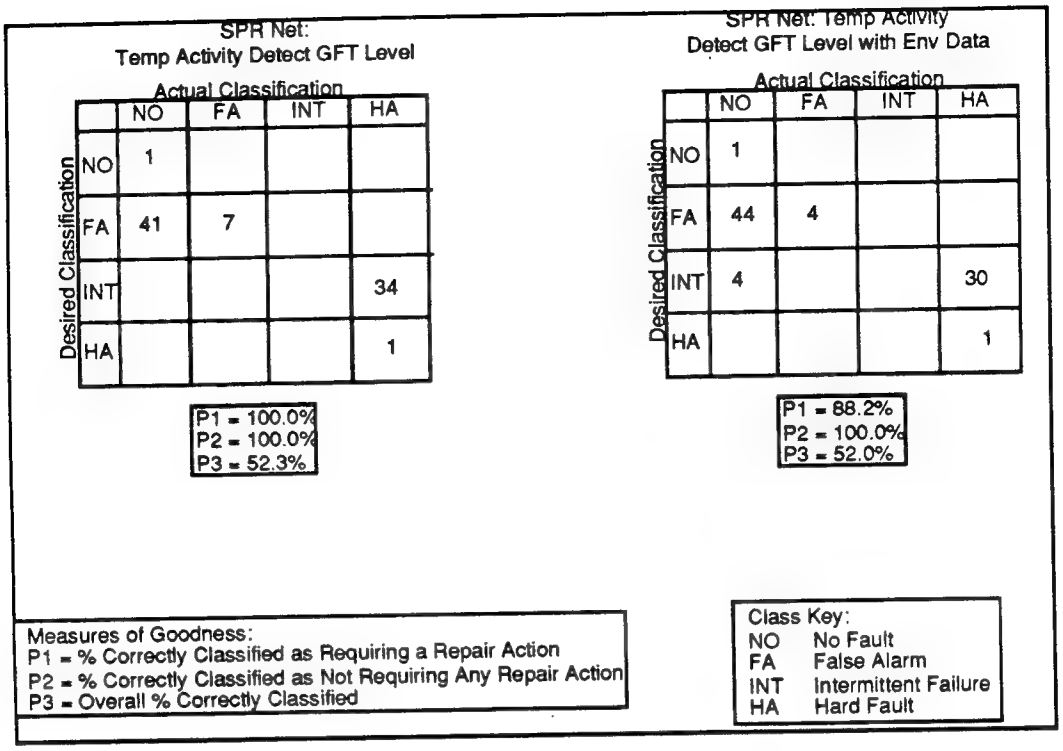


Figure 6.3.1-3. Validation Results Approach 3, GFT BIT Reporting Level

6.3.2 Noise Test Results

Figures 6.3.2-1 through 6.3.2-3 show the results of noise testing for Approach 3. The addition of noise to the test data had little impact on network performance, since all data was classified as either No Fault or the Hard Fault.

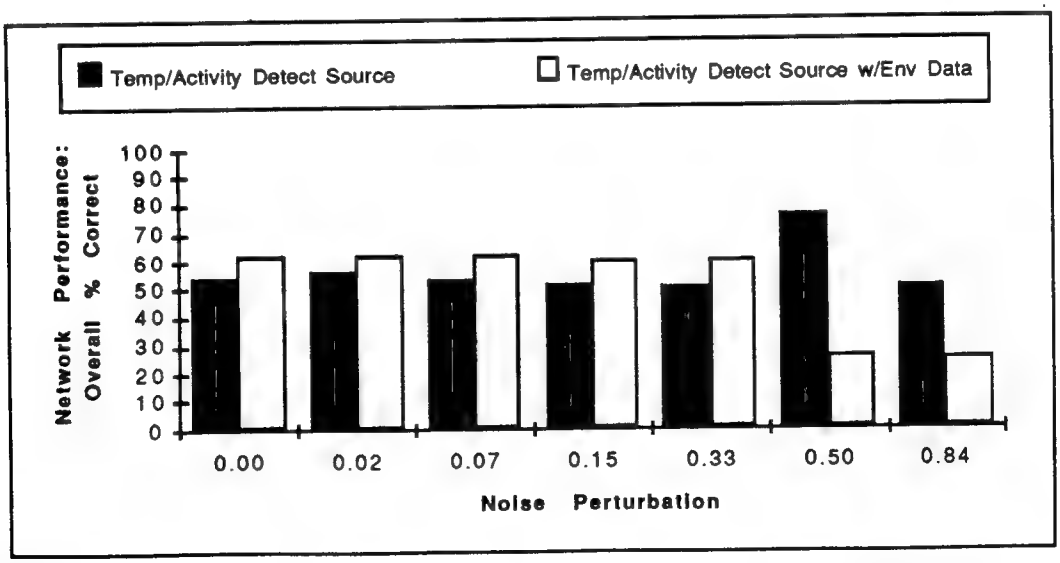


Figure 6.3.2-1. Noise Testing Approach 3, Source BIT Reporting Level

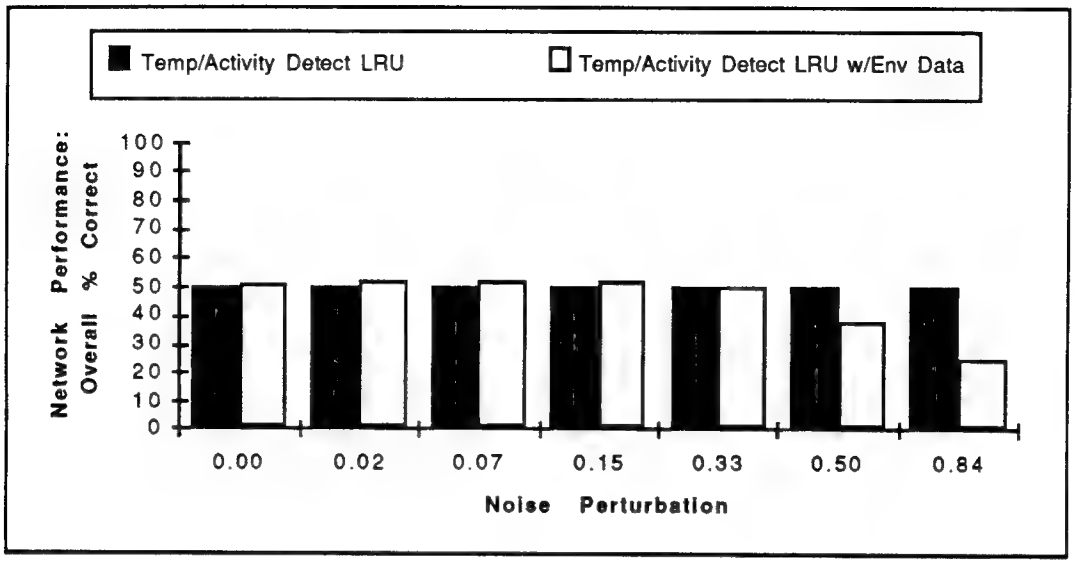


Figure 6.3.2-2. Noise Testing Approach 3, LRU BIT Reporting Level

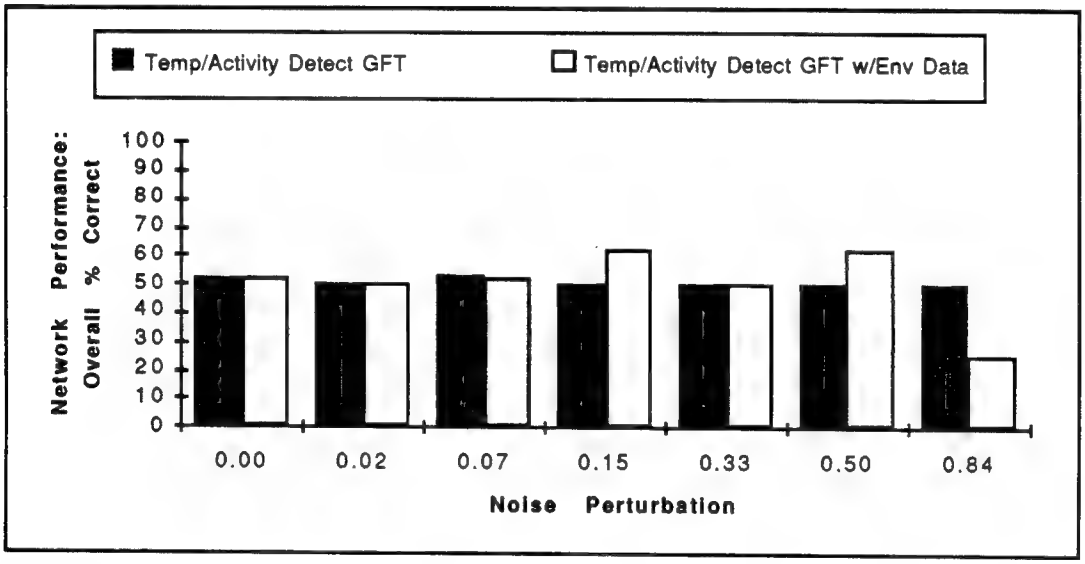


Figure 6.3.2-3. Noise Testing Approach 3, GFT BIT Reporting Level

6.3.3 Summary

The SPR network was the least successful of the three models which were investigated. It was not able to distinguish between the four classifications, at any level of BIT fault reporting. It was able to distinguish between reports which would require a repair action and those which would not. The addition of environmental data to the fault signatures did not improve the network

performance. The performance of the network was slightly better at higher levels of BIT reporting. Noise testing augmented the deficiencies.

We believe that the SPR network failed to perform well because of the large number of network inputs. At the source level, the number of network inputs was 300. The addition of environmental data only increased the number of inputs, and was therefore not of benefit. At the higher levels of fault reporting, the number of inputs was slightly less and the performance of the network was slightly improved. Subsequent discussions with the vendor revealed that the network was typically used with an order of magnitude less inputs (typically 10-20), although they knew of no architectural or computational reason for poor performance with larger numbers.

We concluded that SPR is a poor performer in this problem domain.

6.4 Approach 4 (REINFORCE/Temperature/Viterbi) Results

6.4.1 Validation Test Results

The validation test results for Approach 4 are shown in Figures 6.4.1-1 and 6.4.1-2. Figure 6.4.1-1 presents the results at the source and LRU levels of BIT fault reporting. Environmental data enhancement was not added to the fault report signatures for this approach, because environmental information was used in the calculation of the network reinforcement signal. Figure 6.4.1-2 shows the results at the GFT level. The results are very good. The REINFORCE networks were 100% correct in all classifications at all levels of reporting.

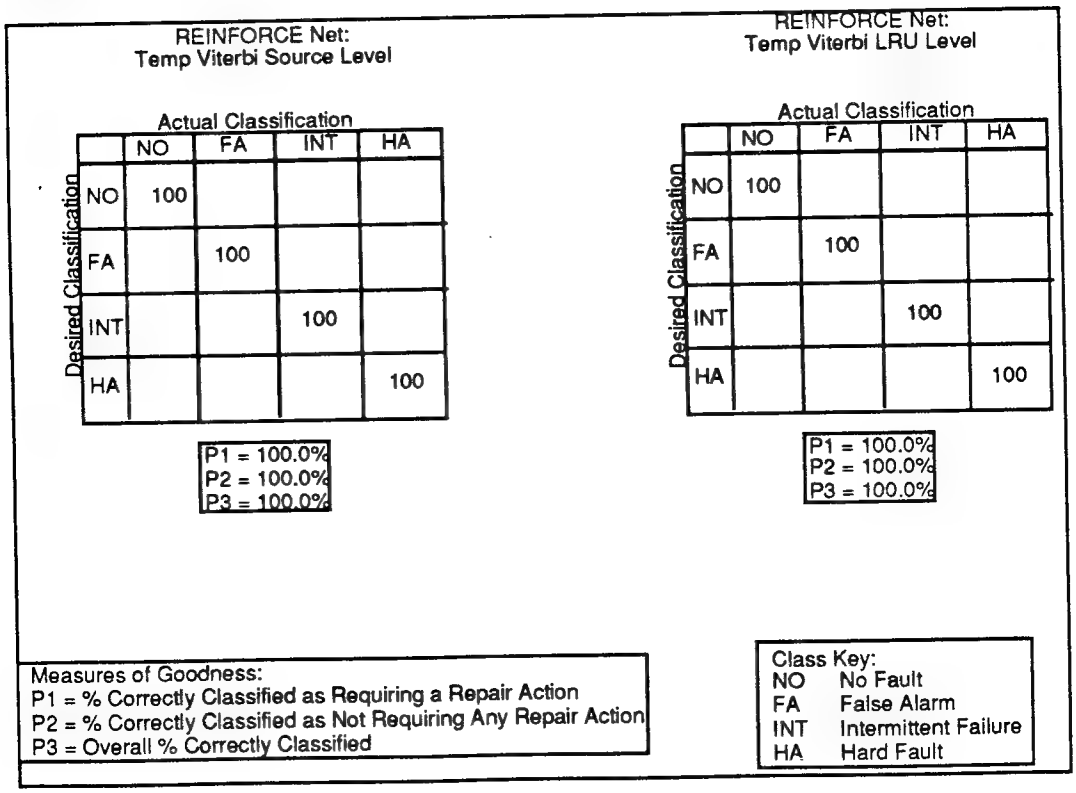


Figure 6.4.1-1. Validation Results Approach 4, Source/LRU BIT Reporting Level

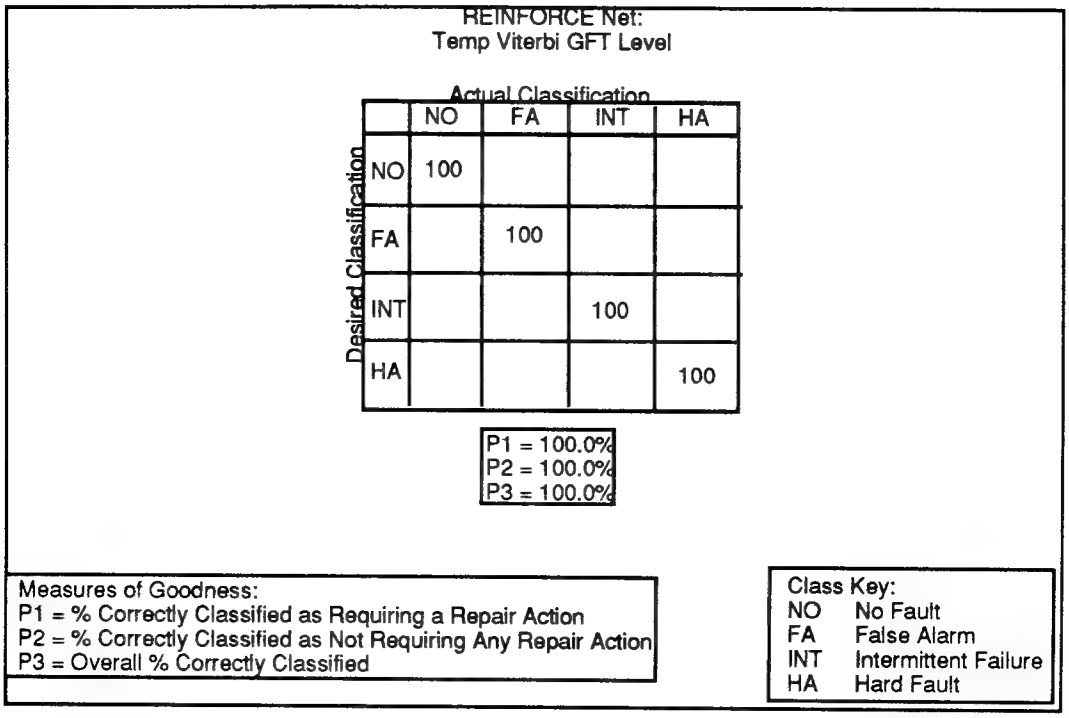


Figure 6.4.1-2. Validation Results Approach 4, GFT BIT Reporting Level

6.4.2 Noise Test Results

Figures 6.4.2-1 through 6.4.2-3 show the results of noise testing for Approach 4. Although the addition of noise to the test data affected network performance, in comparison to the other models, the REINFORCE networks seemed to be the most tolerant of noisy data, especially at the GFT level of reporting. As was seen for all networks, the REINFORCE networks were always able to correctly classify No Faults and Hard Faults.

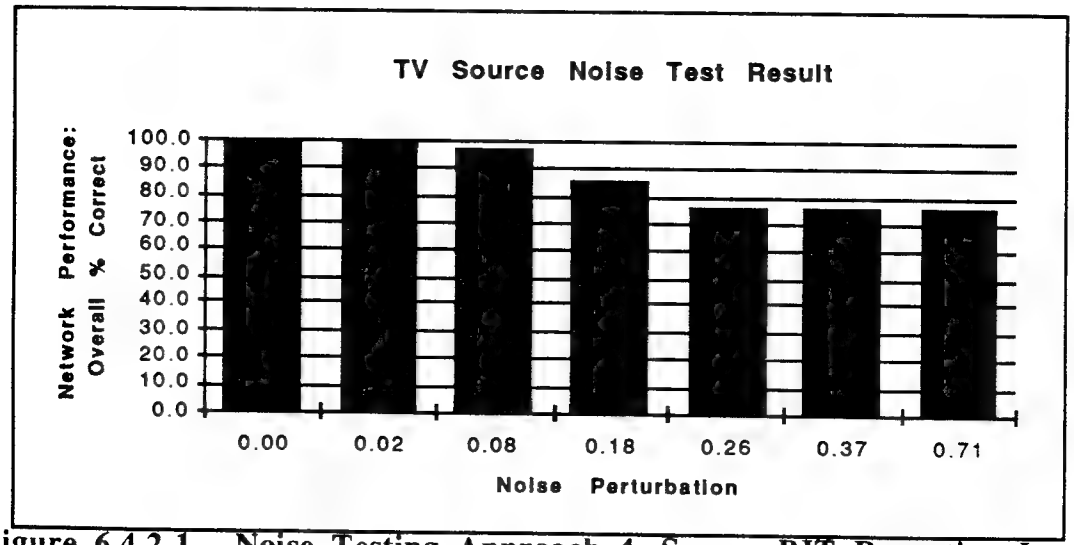


Figure 6.4.2-1. Noise Testing Approach 4, Source BIT Reporting Level

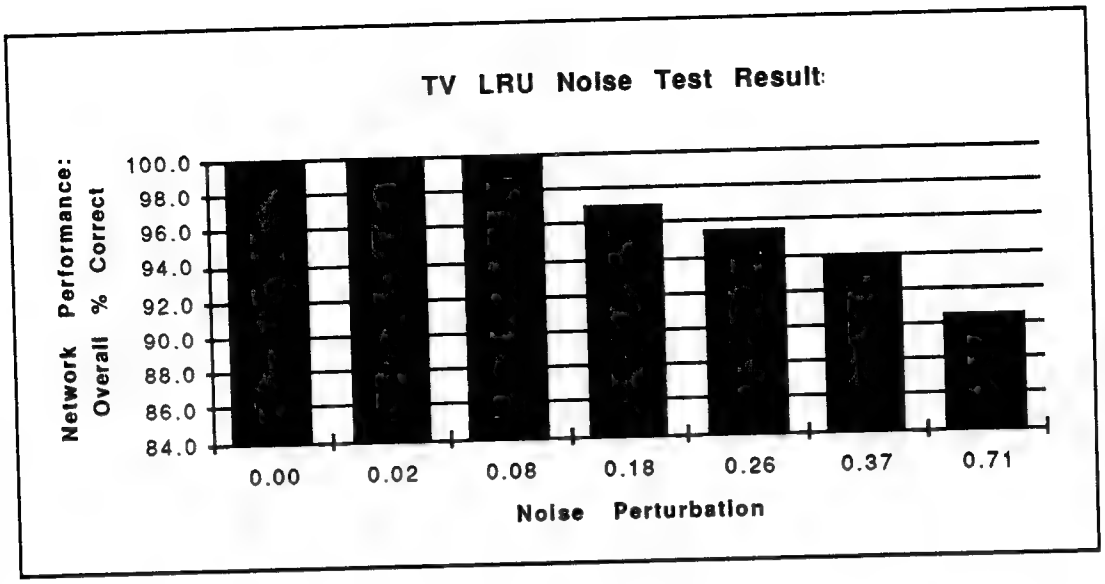


Figure 6.4.2-2. Noise Testing Approach 4, LRU BIT Reporting Level

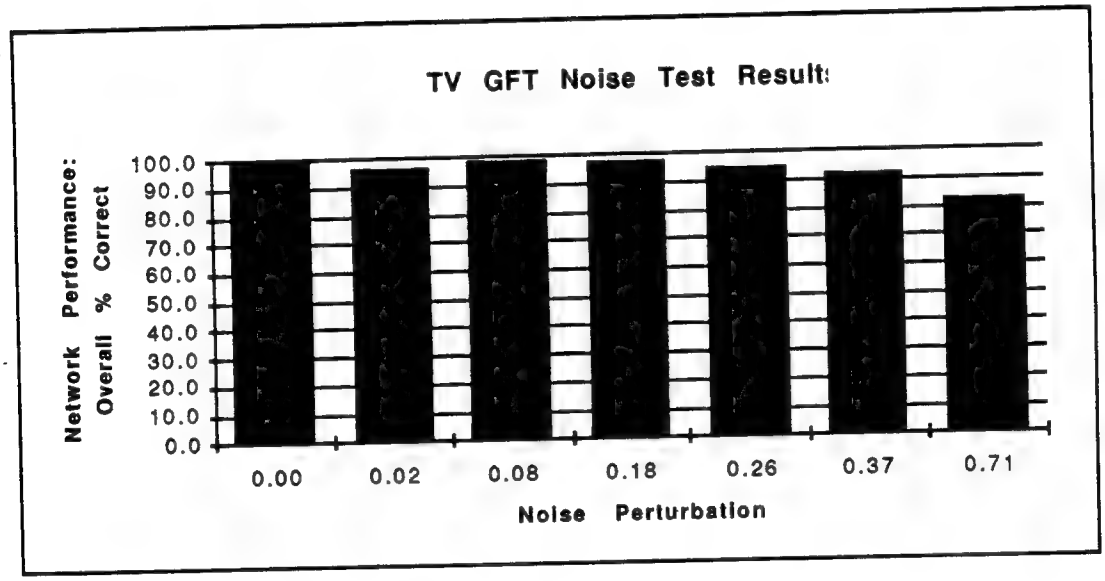


Figure 6.4.2-3. Noise Testing Approach 4, GFT BIT Reporting Level

6.4.3 Summary

The REINFORCE network was quite successful, performing with 100% correct classifications for all validation data at all levels of BIT reporting. We concluded that it is also a viable candidate for future study in this problem domain.

7. IMPACT STUDY

The purpose of the Impact Study was to assess the technical factors involved in the insertion of Neural Net False Alarm Filter technology into communications equipment, and to evaluate the resulting cost/benefit tradeoffs.

The analysis was confined to organizational level maintenance only, i.e., insertion of the technology at the organizational level. Although neural network technology may be effective in intermediate or depot support equipment, this area was not examined. In addition, the Impact Study was only one aspect of the tasks undertaken on the NNFAF contract. As such, resources were limited. Consequently, the cost/benefit models and conclusions must be viewed as first-order approximations on which more detailed analysis could be based.

Finally, due to the widely varying complexity and technologies of communications systems, it proved extremely difficult to draw any specific conclusions regarding neural network false alarm filter insertion without considering specific systems.

7.1 Methodology

The assessment of technical and cost/benefit impact was partitioned into System Components, System Maturity and Neural Network type. The partitioning may be viewed as a three-dimensional matrix, with major system components along one axis, system maturity along a second axis, and neural network type along the third axis (see Figure 7.1-1).

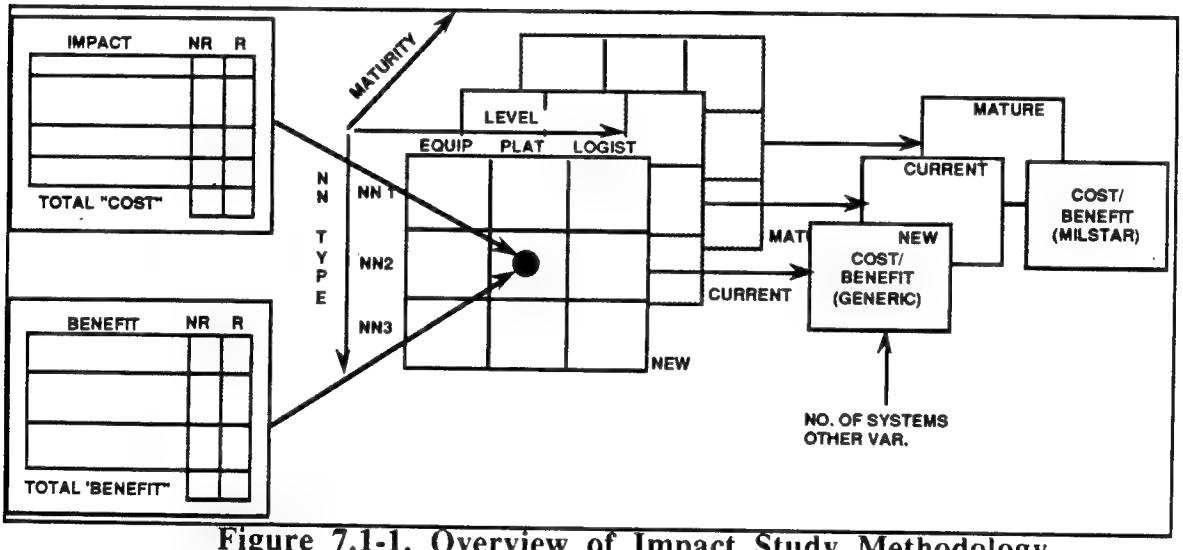


Figure 7.1-1. Overview of Impact Study Methodology

At the system component level, analysis was partitioned into three areas: the prime mission equipment (PME), the equipment platform, and the logistics/support system. Prime equipment includes the hardware and software which comprise the actual mission equipment, exclusive of platform-provided accessories or any O-level support equipment. Communication PME typically consists of RF components, modulator/demodulator subsystems, baseband user interfaces and some type of overall control function with operator interfaces. Impact to the PME may include additional processing and memory, additional BIT sensors, and additional or different software.

The platform area examines impact to the platform on which the PME is mounted. Platform impact typically may include increased power, volume, weight and cooling requirements, and possibly additional wiring to connect to platform-based sensors (e.g., environmental measurement) which support the neural networks.

The logistics and support system consists of the intermediate and depot support facilities, the spares supply system and related transportation and tracking methods, technical manuals and related procedures, maintainer personnel and training, etc. Although the effects of implementing neural networks at the maintenance facility level were not examined, one of the major savings resulting from reduced false alarms could accrue from reducing the occurrence of unnecessary removals which lead to costly Re-test OK situations at the lower maintenance levels.

Recognizing that the cost and benefit of inserting NNFAF technology may vary considerably with the age of a system, the system maturity dimension was partitioned into three categories: Mature, Current and Future. Mature systems are those which have typically been in the field and in use for more than seven or eight years. These systems are usually not highly processor-intensive, and likely were designed to less stringent BIT requirements than more recent systems. As a result, they may have insufficient BIT detector coverage and processing capacity to effectively implement neural network false alarm filtering without major expenses.

Current systems are those which are through the design and testing phase and are just starting to be procured in quantity and fielded. Typically these systems are processor intensive and have reasonably good BIT coverage as a result of more stringent requirements. Since they have a relatively long life remaining, they are also likely to have opportunities to "piggyback" the neural network false alarm filter insertion on other upgrades.

Future systems are those which are just entering the design phase or are yet to be specified and designed. These systems are likely to be highly processor-intensive. However, they also offer the opportunity to incorporate neural network false alarm filtering techniques early in the design process, rather than as a modification to existing hardware and software.

The third dimension of partitioning encompassed the three neural network models (Backpropagation, REINFORCE and SPR) which were evaluated in this program. These models are described in Section 5.4. The network model or type influences throughput and memory burden and the expected improvement in False Alarm Reduction (FAR). The former influences the cost, while the latter influences the potential savings.

In order to evaluate the impacts of NNFAF , a system of three linked Excel spreadsheets was developed. The Cost Model spreadsheet lists all the identified factors which may affect the cost of inserting neural network filtering technology into a system, and allows the entry of cost estimates against each factor. The Benefit Model spreadsheet prompts for the system parameters and factors which influence the expected savings resulting from implementing neural network false alarm filtering in a system. Both the cost and benefit sheets encompass the PME, Platform and Logistics domains. A third spreadsheet uses the cost and benefit data to graphically display the resulting trade-off space. The models and spreadsheets are described in detail in Section 7.2 below. An example of the Cost/Benefit Display spreadsheet is shown in Appendix H.

As previously mentioned, the cost and benefit spreadsheets encompass the "system component" dimension of the matrix (see figure 7.1-1) by permitting entries categorized by PME, Platform and Logistics. Beyond that, the spreadsheets are not specifically tailored to the three maturity levels or the three neural network types. Instead, these categories influence the cost and benefit entries.

Hence a compete analysis of all points in the matrix would result in a set of three spreadsheets for each of the nine rows contained in the (NN type X Maturity) space of the matrix. However, it was deemed not necessary to perform any analysis for the SPR algorithm due to its poor performance. The REINFORCE and Backpropagation results are very similar, but there are differences in computational burden which will affect implementation cost. As an example of a specific current system, an analysis of the Air force MILSTAR Terminal system was performed, based on actual MILSTAR technical and logistics data.

7.2 Models

7.2.1 Cost Estimation Model

The cost model/spreadsheet prompts for cost estimate inputs associated with achieving a baseline FAR, typically the FAR already known to be achieved by the system. Using the number of systems (N) and a pair of complexity factors, the model estimates the costs at several points, spanning an order-of-magnitude improvement (lowering) of the FAR. The cost model and the cost spreadsheet partition the cost parameters associated with neural network false alarm filter insertion into four major categories.

The first category consists of those cost factors which are essentially independent of incremental changes in the desired FAR and also independent of the number of systems (N), i.e., non-recurring costs. The second category consists of components which are still independent of FAR, but depend linearly on N (recurring costs). The third category includes costs which are expected to escalate with decreasing FAR, but are independent of N. The fourth category comprises costs which escalate with decreasing FAR and have a linear dependence on N. Viewed another way, the total estimated cost associated with a given FAR has a non-recurring component, part of which is independent of the FAR and part which varies with FAR, and a recurring component, part of which is independent of the FAR and part which varies with FAR. The total cost associated with a given FAR is the sum of these four components.

The cost growth factor as a function of decreasing FAR should have several characteristics. It should be equal to 1.0 when the FAR equals the baseline, or initial FAR; it should asymptotically approach infinity as the FAR approaches zero; and it should provide for a complexity factor which affects the growth rate. A simple growth function which meets these criteria is

$$G = (F_i/F)^K \qquad \text{(Eq. 7-1)}$$

where

G is the growth factor
F_i is the initial or baseline FAR
F is the FAR variable, $F_i \geq F \geq 0$
K is the complexity factor

Figure 7.2.1-1 shows the growth factor G as a function of changing F and K for an F_i of 20%.

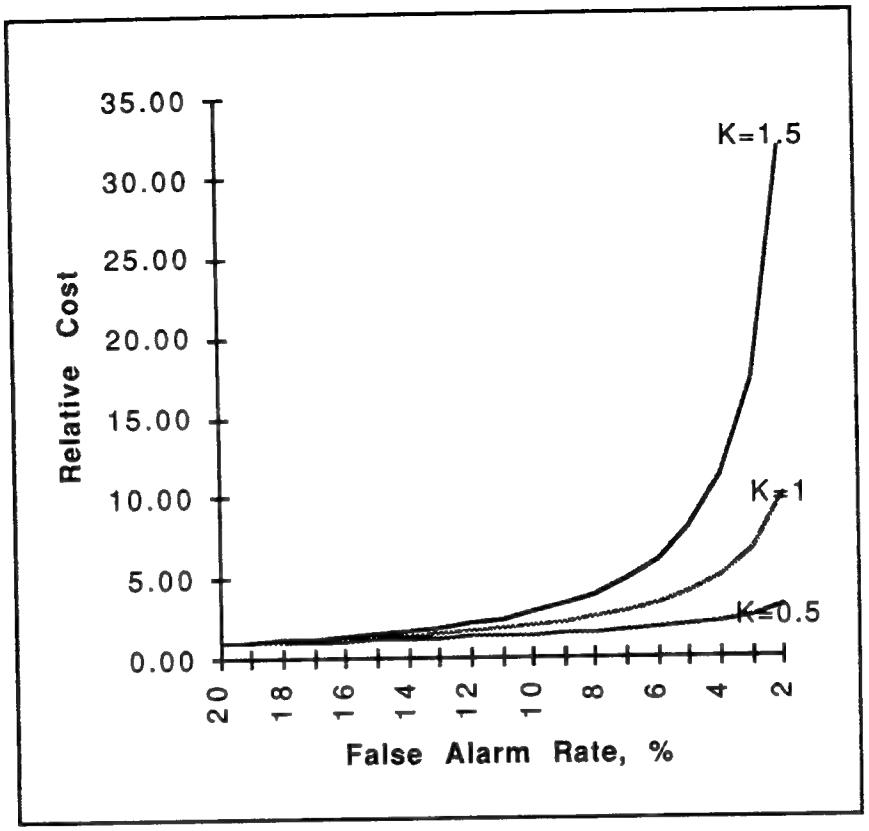


Figure 7.2.1-1. Cost Growth Factor as a Function of F and K

Using the growth factor results in an equation for the total cost as a function of N, F, F_i, and K as follows:

$$C(N,F,F_i,K1,K2) = S1 + NS2 + S3(F/F_i)^{K1} + NS4(F/F_i)^{K2} \qquad \text{(Eq. 7-2)}$$

where

C is the total cost
N is the number of systems
S1 is the sum of the costs independent of both FAR and N
S2 is the sum of the costs independent of FAR but dependent on N
S3 is the sum of costs dependent on FAR but independent of N
S4 is the sum of costs dependent on both FAR and N
K1 is the complexity factor associated with S3
K2 is the complexity factor associated with S3
F is the FAR variable
F_i is the initial (baseline) FAR

The two complexity factors, K1 and K2, are intended to allow tailoring of the spreadsheets to a wide variety of systems. Ideally, selecting appropriate values of K1 and K2 would be based on regression analysis of historical upgrade costs across a wide variety of system types. However, this type of task is well beyond the scope of the NNFAF contract. For the present, engineering

judgment is used to select values. K1 and K2 are user-entered parameters to facilitate future applications of the spreadsheet.

Figure 7.2.1-2 shows an example of the individual (S1, S2, S3 and S4) cost curves and the resulting composite curve. The costs entered into the spreadsheet represent estimates to achieve a FAR of F_i. The positive Y-intercept shown in Figure 7.2.1-2 highlights the fact that there is a "cost of admission" to introduce neural network technology to a point which achieves the level of performance already available by some other means already in the system. For future systems this "cost of admission" will be minimal.

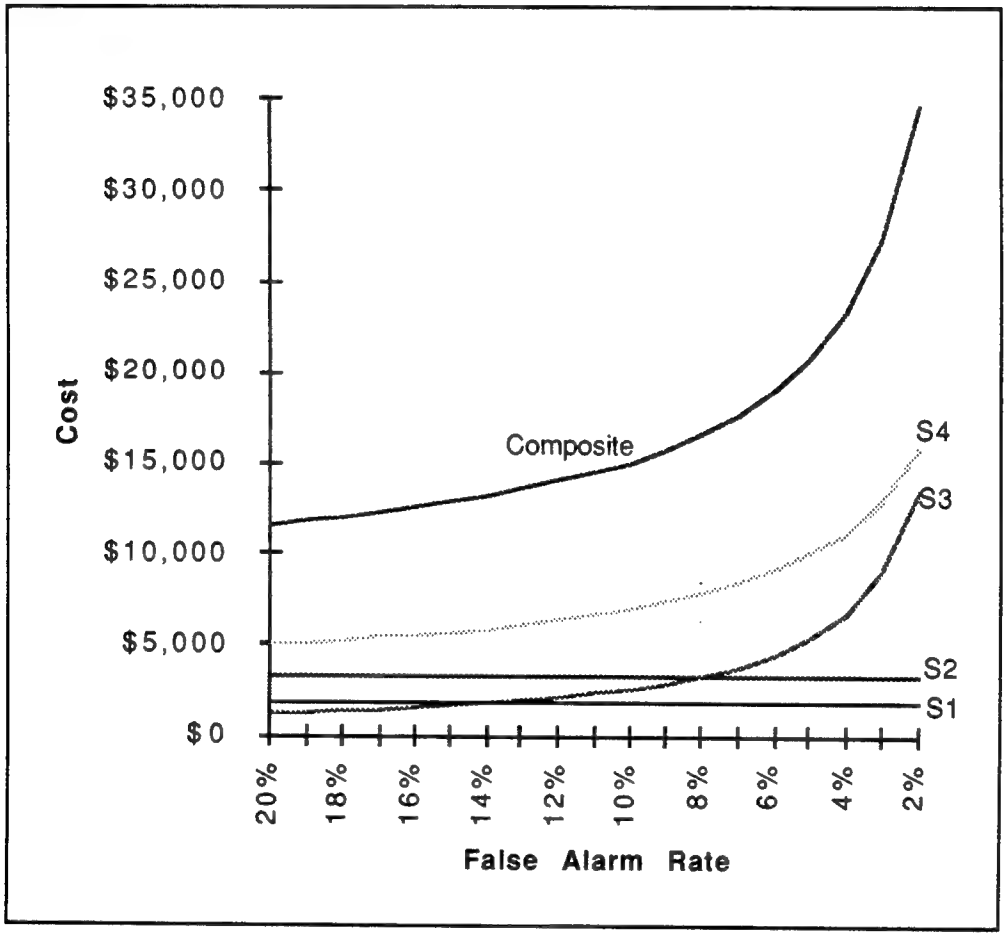


Figure 7.2.1-2. Individual and Composite Cost Curves

Based on the inputs and the model, the cost spreadsheet output is a set of costs for a range of FAR between F_i and $0.1F_i$ (i.e., one order of magnitude) at intervals of $0.1\ F_i$. This set of costs is an input to the Cost/Benefit Display Spreadsheet (see Section 7.2.3).

7.2.1.1 Processing and Memory Issues

One of the key factors which will affect both hardware and software cost estimates is the processing throughput and memory needs associated with implementing neural network filters. Therefore this section provides some guidelines for estimating processing and memory

requirements. Note that in general it is necessary to perform the neural network processing in real-time while the system is running and performing all normal on-line tasks.

Figure 7.2.1.1-1 shows the configuration assumed for the ith BIT detector for throughput and memory estimation.

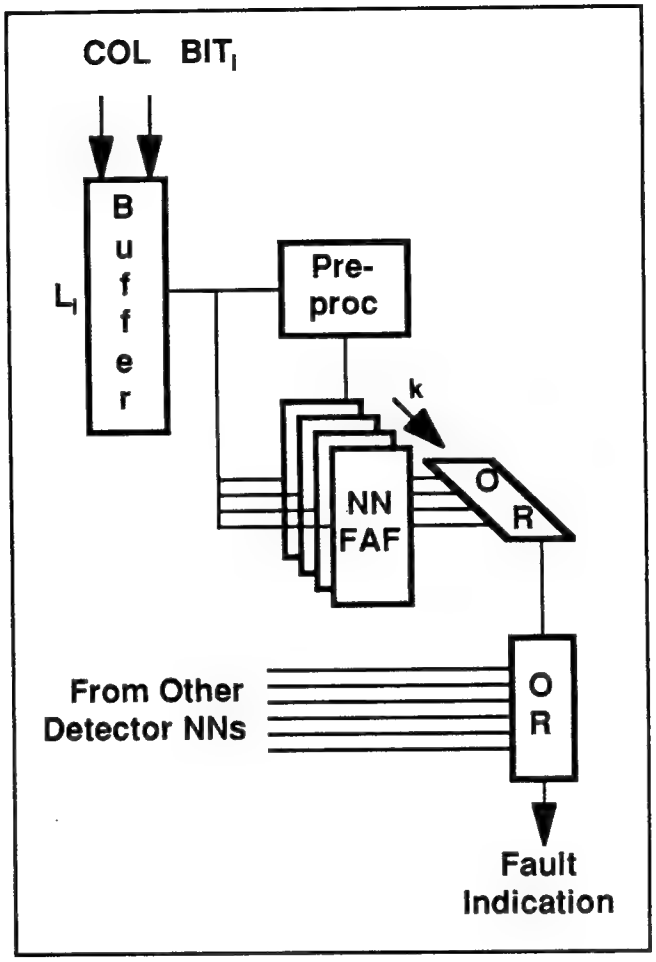


Figure 7.2.1.1-1. NNFAF Processing/Memory Burden Model

During system operation, each bit detector (and associated collateral data such as environmental information) is accumulated in a buffer of length L_i, where L_i corresponds to the length of the signature appropriate to the selected neural network model. In a straightforward approach, every L_i samples, a set of k neural network filters process the signature and determine whether there is a reportable fault. The k filters are needed because it is unlikely that one neural network can be trained to work effectively across a wide variety of false alarm signatures generated by differing causes. In effect, there is a bank of filters, each tuned to a different signature. For a given BIT detector, the outputs of the filters are ored across the set of filters, then ored with the accumulated outputs of filters associated with other detectors in the system to produce a fault report.

This approach requires that all neural network filters in a system be running in real time during system operation. In large systems with many detectors, the resulting processing burden becomes

intractable. A more realistic implementation provides a pre-processor function for each detector which examines the buffer to determine whether it is necessary to perform false alarm filtering. This preprocessor may be as simple as detecting one or more positive fault indications in the buffer. If a fault indication is detected, the updating of the remaining buffers in the system is stopped to retain the entire state of the system fault status, and the neural network processing is invoked only for those buffers which have a fault indication present.

With the above approach, during normal, non-faulted system operation, none of the neural network processing needs to run. More importantly, in most cases in the presence of a system problem, only a small portion (e.g., less than 10 %) of the detectors will report a fault. Therefore, only those filters which are necessary are invoked, minimizing the neural network processing burden. In addition, because the system BIT status is "frozen", the necessary neural network processing needs not execute in real time, but only needs to be completed within a specified fault latency time. Fault Latency, T_{lat}, is the time from the occurrence of a fault until it is reported to the user. A reasonable T_{lat} is highly application dependent. For example, an Identify-Friend/Foe (IFF) system should have very short T_{lat}, typically less than two seconds. Communication systems can generally afford higher T_{lat}, on the order of 30 seconds or more.

Note that if an overwhelming number of detectors have tripped, there is a high probability that the system is no longer operating, in which case allocating system processing to false alarm filtering has minimal impact.

Clearly, processing burden will be highly dependent on the nature of the processing available. In general, the filtering will probably be done by software, although hardware integrated circuit implementations of Backpropagation are available (this subject is discussed further on in this section). In any case, the burden estimates are most conveniently normalized to a primitive operation. In the case of neural networks, the predominant operation is a multiply followed by an add, so multiply/add cycles are a convenient primitive.

For a given neural network, the number of Multiply/add cycles is approximated by

$$M_{ik} \approx L_i H_{ik} \qquad (\text{Eq. 7-3})$$

where

M_{ik} is the number of multiply/adds associated with the kth filter of the ith BIT detector
L_i is the number of samples in the buffer of the ith BIT detector (this also equals the number of input nodes)
H_{ik} is the number of hidden nodes in the kth neural network of the ith BIT detector

This equation neglects the operations to compute the output nodes because the number of output nodes (4) is typically only a small fraction of the number of input nodes, which dominate the calculation. The network transfer function (typically sigmoid) calculations need to be performed once per hidden node, and would likely be implemented via pre-computed table look-up techniques. Therefore the sigmoid calculations are also neglected.

Using the equation above, the total throughput may be estimated by

$$R_{tot} = P \sum_i \sum_k M_{ik} / T_{lat} \qquad (\text{Eq. 7-4})$$

where

R_{tot} is the total Multiply/Add rate in operations per second
P is the fraction of BIT detectors which are likely to indicate faults simultaneously
M_{ik} is the number of Multiply/adds associated with the kth filter of the ith BIT detector
T_{lat} is the maximum permissible fault latency time

To assess the order of magnitude of the processing burden, consider an example system with 1200 BIT detectors, with an average of five neural network filters per detector. The filters average seven hidden layer nodes and 300 input nodes each (typical values). P is assumed to be 0.1, and T_{lat} is five seconds. R_{tot} is calculated to be 252,000 operations per second.

R_{tot} estimates the burden independent of the technology. The translation of R_{tot} into Millions of Instructions per Second (MIPS) requires knowledge of the host processors. Modern processors may require only two or three machine instructions per multiply/add cycle, yielding a requirement well under one MIPS. Older technology, especially if hardware support for multiplication is not available, may well need hundreds of machine instructions for each operation, leading to a requirement on the order of 25 MIPS. The processing need not be performed in one processor, of course, and typically would be distributed among several processors in a system.

The ith BIT detector requires enough RAM to store a signature Li bits in length. Summing over the BIT detectors yields the required memory in bits. For the above example, the memory requirement would be 360,000 bits or 45 Kbytes. This does not represent an extraordinary memory demand given recent RAM technology.

RAM is also required to provide working storage as the network node values in each layer are calculated. For the Backpropagation network, however, as the hidden nodes are traversed, it is never necessary to store more than L_i intermediate results. (The REINFORCE network used no hidden nodes). Furthermore, as soon as the output of one network is calculated and stored, the working storage may be used for the next network. Therefore, working storage requirements are limited to the maximum value of Li, at (typically) two to four bytes per location. This will be negligible compared to other memory requirements.

The dominant factor in storage requirements is the memory to store the network connection weights. In the extreme case, the multiply portion of each multiply/add for each network may utilize a unique weight value. Hence, the total connection weight storage is equal to the total number of multiply/add operations. Note that unlike the calculation for the multiply/add rate, which assumed that only a fraction (P) of the filters would have to be executed, it is necessary to store all connection weight values. Hence the number of weights (N_W) is

$$N_W = B \sum_i \sum_k M_{ik} \qquad \text{(Eq. 7-5)}$$

where

N_W is the total storage requirement for connection weights in bytes
B is the number of bytes per weight (two is generally sufficient)
M_{ik} is as defined in equation 7-3

For the example system, and assuming 2 bytes per weight, N_W is about 25 Megabytes. In all probability these weights would be stored in some form of Erasable, Programmable Read-Only memory (EPROM), albeit distributed through the system. While well within the bounds of present technology, this represents a significant amount of PROM, along with related power supply and cooling considerations. Magnetic storage is a possibility if system environmental and reliability constrains permit. However, access times associated with use of magnetic storage must be considered in the throughput calculations.

Application-Specific Integrated Circuits (ASICs) which implement the Backpropagation algorithm are available from several manufacturers, and offer an alternative approach to inserting NNFAF technology. Throughput considerations indicate that each of these devices could support from approximately ten to fifty detectors. Therefore, a reasonable implementation might put one or two devices in each LRU or subsystem, with the outputs feeding a centralized processor which reduces the outputs to operator/maintainer messages.

The use of such devices eliminates the need for the neural network "simulation" software, along with the associated development or procurement costs. In addition the devices are fast enough that real-time processing on-line would be possible, eliminating the need for pre-processing to determine which filters should run.

However, there are a number of drawbacks. Insertion of additional distributed hardware raises replication costs while negatively affecting power, volume and reliability. This would be a strong consideration if a system already had sufficient reserve processing to implement software-based algorithms, or if there were a large number of systems involved. Given the rate of change in technology for ASICs, processors, memory, and neural networks, the suitability of hardware- versus software-based implementations must be considered on a case by case basis.

7.2.1.2 Cost Model Inputs

Table 7.2.1.2-1 lists the specific cost items identified and provides a brief explanation of each item. Note that some items allow for entries in more than one category (e.g., both independent of FAR and dependent on FAR) to accommodate situations where the cost may have both a dependent and an independent component. Also, the list of items is intended to be comprehensive and generally applicable to any system. Some specific systems may have zero cost associated with one or more of the items, depending on the system.

Table 7.2.1.2-1. Cost Spreadsheet Inputs

Input Parameter	Description
EQUIPMENT	Costs captured in this section are specific to the host system and exclude costs to the platform or logistics system.
Engineering Feasibility Study	Non-Recurring Engineering (NRE) (Government) This study will determine the efficacy of a given approach to implementing NN. Average study runs from 6 to 12 MM in effort. Depending on the scope of the effort, will usually fall into the Cost to Implement NN category.

Table 7.2.1.2-1. Cost Spreadsheet Inputs (continued)

Input Parameter	Description
Define Requirements	NRE (Government) to define what equipment changes have to be made in order to accommodate the NN approach.
Issue RFP and Select Contractor	The NRE (Government) cost to create an initial RFP, advertise the contract and collect/evaluate proposals, including the final selection and kick-off.
Select NN Model	This NRE may be partially covered in the proposal process. It is the cost associated with performing trade-off studies for the various NN approaches.
Design H/W Equip Mods to NN Host	NRE for designing system HW changes required to implement NN. There may be a component which involves the additional costs (i.e. sensors, new processors, memory) to improve the FAR.
Design S/W Mods to NN Host	NRE to design or modify the SW on the host system to incorporate NN. There may be a component which involves additional SW for sensors etc. to improve the FAR.
Design, Code and Test NN Model	The NN will require developing SW tailored for each application. This NRE is the cost to develop (or acquire), code and test the SW required. The bulk of this cost is included in implementing NN, since few additional costs are needed for improvement of FAR.
Write Test Plans and Procedures	NRE to cover the cost of developing HW and SW Test Plans and Procedures and any reviews.
Generate NN Training Data	NRE to generate the initial data used to train the NNs. Initial data will likely require extensive analysis by system experts.
Training NN (initial)	NRE labor for training the NNs using the generated data.
Procure New Hardware	This is the cost of procuring any new hardware for the host system to accommodate the NN. There may be both NRE and recurring components. Mostly specific to improving FAR.
Procure Spares for New Hardware	The per system cost to purchase spares for new hardware. Considered initial NN startup cost, but may have FAR improvement costs.
Install H/W Mods to Incorporate NN	The per system cost to install new and/or modify existing system hardware to incorporate NN.
Install S/W Mods to Incorporate NN	The per system cost to install or modify existing system software to incorporate NN.
Install NN Software	The per system cost to install the NN software.
Update Existing S/W Documentation	Self explanatory.

Table 7.2.1.2-1. Cost Spreadsheet Inputs (continued)

Input Parameter	Description
Update Existing H/W and System Documentation	Self explanatory.
Prototype Integration and Test	The NRE cost to perform integration & initial testing of the BIT system with the NN incorporated.
Regression Testing	The NRE cost to ensure that the original system capabilities have not been adversely impacted by installing NN.
Acceptance Test for Each System	Recurring cost of acceptance test for each system.
Sell-Off Test	NRE costs to perform sell-off tests of prototype system against revised specifications.
PLATFORM (ANCILLARY IMPACT)	Costs captured in this section relate to platform changes to accommodate the modified prime equipment.
Site Survey	NRE cost to assess platform changes necessary for installation of updated system.
Design Platform Modifications (volume, power, cooling, weight and balance, safety, etc.)	NRE cost to design modifications to the existing host platform for NN implementation.
Arrangements for Access to Platform	Recurring cost to get the platform to a place where modification and installation of the equipment can be performed. Applies only if installation cannot be performed as part of or in parallel with some other modification.
Platform Modifications (System Specific)	Per System costs to alter the platform to accommodate NN within the host system.
Installation Verification	NRE Cost to verify the installation is correctly performed.
Update Platform Documentation	NRE Cost to update technical orders, manuals, and other changes to the platform as a result of NN.
Regression Tests	NRE cost to ensure the platform ancillary systems were not adversely affected by the installed modification.
PLATFORM (NN SPECIFIC)	Costs captured in this section relate to platform changes to accommodate NN equipment outside the prime mission equipment (e.g., platform environmental sensors).
Site Survey	NRE cost to assess platform changes necessary for installation.
"Design Platform Modifications (volume, power, cooling, Weight and Balance, safety, etc.) "	NRE cost to design modifications to the existing platform specific to the host system for NN implementation.

Table 7.2.1.2-1. Cost Spreadsheet Inputs (continued)

Input Parameter	Description
Arrangements for Access to Platform	Cost to get the platform to a place where modification and installation of the equipment can be performed. Applies only if the cannot be performed as part of or in parallel with some other modification and not already covered above.
Platform Modifications (NN Specific)	Per System Costs to accommodate NN external to the host system .
Installation Verification	NRE Cost to verify the installation is correctly performed.
Update Platform Documentation	NRE Cost to update technical orders, manuals, and other changes to the platform as a result of NN.
Regression Tests	NRE cost to ensure the host system was not adversely affected by the installed modification.
LOGISTICS	This section identifies the various support tasks and logistics disciplines which may be required to support NN.
Analyze Equipment Mods for Logistic Concerns (Reliability, LSA, etc.)	Safety, reliability, maintainability, maintenance procedures modifications, human factors analysis, etc. are all covered in this NRE cost.
Specify and Procure System for Training/Retraining of NN.	The cost for procuring computer hardware and software required for gathering maintenance data, creating new NN parameters, and distributing the changes.
Plan and Install Mechanism to Collect and Analyze Historical Data	Plans and installation costs for equipment and procedures procured an previous step. Includes all testing, verification, validation, etc.
Arrange Access to Equip to Perform S/W Update	Per System Cost to perform the update to the NN software.
Collect Historical Data for Re-training	This is the lifetime cost per system for periodically sending the fault data to the central data bank for processing. Assumed to be more often during first five years, and then tapering off.
NN Retraining (Periodic)	Costs associated with analyzing collect data and periodically generating and distributing revised Neural Nets.
Distribute Retrained Networks	This is the lifetime cost per system of distributing the retrained networks to each system Assumed to be more often during first five years, and then tapering off.
Technical Order Updates	Cost of updating the technical manuals for operator and maintenance procedure changes, part number changes, and other documentation not already covered.

Table 7.2.1.2-1. Cost Spreadsheet Inputs (continued)

Input Parameter	Description
Modify Training Courses	NRE cost to accommodate any changes required in existing or developing training courses as a result of NN installation and modifications to the platform.
Modify Support Equipment	NRE cost of Organizational Support Equipment changes required as a result of NN implementation. Peculiar Support Equipment or Special Support Equipment changes should include Support Equipment Recommendation Data (SERD) costs, Logistics Support Analysis costs and the costs for verification/validation of procedures and tech manuals for that item.

7.2.1.3 Cost Model Outputs

The output of the cost worksheet is a set of ten cost values representing the total costs for F_i and nine additional values of F down to $0.1F_i$, calculated using K1, K2 and N in accordance with equation 7-2.

7.2.2 Benefit Estimation Model

The primary quantifiable benefits associated with introducing NNFAF technology accrue from the reduced unnecessary maintenance actions and unnecessary mission aborts resulting from an improved FAR. The benefit spreadsheet therefore prompts for parameters which affect the total expected number of failures over the system life and the costs associated with maintenance actions and mission aborts. These parameters include, among others, the system Mean Time Between Failures (MTBF), the total number of installed systems, the cost of a maintenance action and the cost of a mission abort. The savings is estimated based on reduced unnecessary repair actions ("false pulls") and mission aborts as the FAR improves.

The savings attributable to reduced false pulls is

$$B_{fp} = (N_{Fi}-N_F)\, A_f\, C_f \qquad \text{(Eq.7-6)}$$

where

B_{fp} is the benefit (savings) resulting from reduced false pulls in dollars
N_{Fi} is the number of false alarms at FAR Fi (the baseline FAR)
N_F is the number of false alarms at the (improved) FAR F
A_f is the fraction of false alarms which result in an unnecessary remove/replace (false pull)
C_f is the cost (in dollars) of a false pull

The generally accepted definition of False Alarm Rate is

$$F = N_F / (N_F + N_{ra}) \quad \text{(Eq. 7-7)}$$

where

F is the false alarm rate
N_F is the number of false alarms at FAR rate F
N_{ra} is the number of real alarms

Solving equation 7-7 for N_F yields

$$N_F = N_{ra}\, F / (1-F) \quad \text{(Eq. 7-8)}$$

Similarly,

$$N_{Fi} = N_{ra}\, F_i / (1-F_i) \quad \text{(Eq. 7-9)}$$

Substituting equations 7-8 and 7-9 into equation 7-6 results in

$$B_{fp} = N_{ra}\, A_f\, C_f\, (F_i-F) / (1-F_i)(1-F) \quad \text{(Eq. 7-10)}$$

The total number of expected real alarms over the life of a system, N_{ra}, may be estimated as

$$N_{ra} = (8760\, D\, Y\, N) / T_m \quad \text{(Eq. 7-11)}$$

where

N_{ra} is the total number of expected real failures over the life of a system
D is the duty cycle, the average fraction of time the equipment is on
Y is the expected remaining equipment life in years
T_m is the Mean Time Between Failures in hours
8760 is the number of hours in a year

Substituting equation 7-11 into equation 7-10 yields

$$B_{fp} = 8760\, D\, Y\, N\, A_f\, C_f\, (F_i-F) / T_m\, (1-F_i)(1-F) \quad \text{(Eq. 7-12)}$$

Based on equation 7-12, the benefit (savings) associated with reduced mission aborts due to false alarms is

$$B_{ma} = 8760\, D\, Y\, N\, A_m\, C_m\, (F_i-F) / T_m\, (1-F_i)(1-F) \quad \text{(Eq. 7-13)}$$

where

B_{ma} is the benefit (savings) due to reduced mission aborts
A_m is the fraction of false alarms which result in a mission abort
C_m is the cost (in dollars) of a mission abort

The spreadsheet model estimates a total benefit (savings) by summing the individually identified benefits

$$B_{tot} = B_{fp} + B_{ma} + B_{on} + NB_{or} \quad \text{(Eq. 7-14)}$$

where

B_{on} is a term to include any non-recurring savings not accounted for by B_{fp} and B_{ma}
B_{or} is a term to account for any recurring savings not accounted for in B_{fp} and B_{ma}

B_{on} and B_{or} are terms in the sum to include other quantifiable savings which are not attributable to reduced false pulls and/or reduced mission aborts. These savings are usually much smaller than B_{fp} and B_{ma}, and may include such items as reduced spares (future systems only), reduced training, fewer tech order updates, etc. No attempt is made to explicitly identify the contributors to Bon and B_{or}, but provision is made on the spreadsheet for entry of these quantities to accommodate cases where such savings may be explicitly identified.

Figure 7.2.2-1 is an example of individual and composite benefit curves for a FAR range of 20% to 2%. In the Figure, B1 represents B_{fp}, B2 represents B_{ma}, and B3 represents the sum of B_{on} and B_{or}.

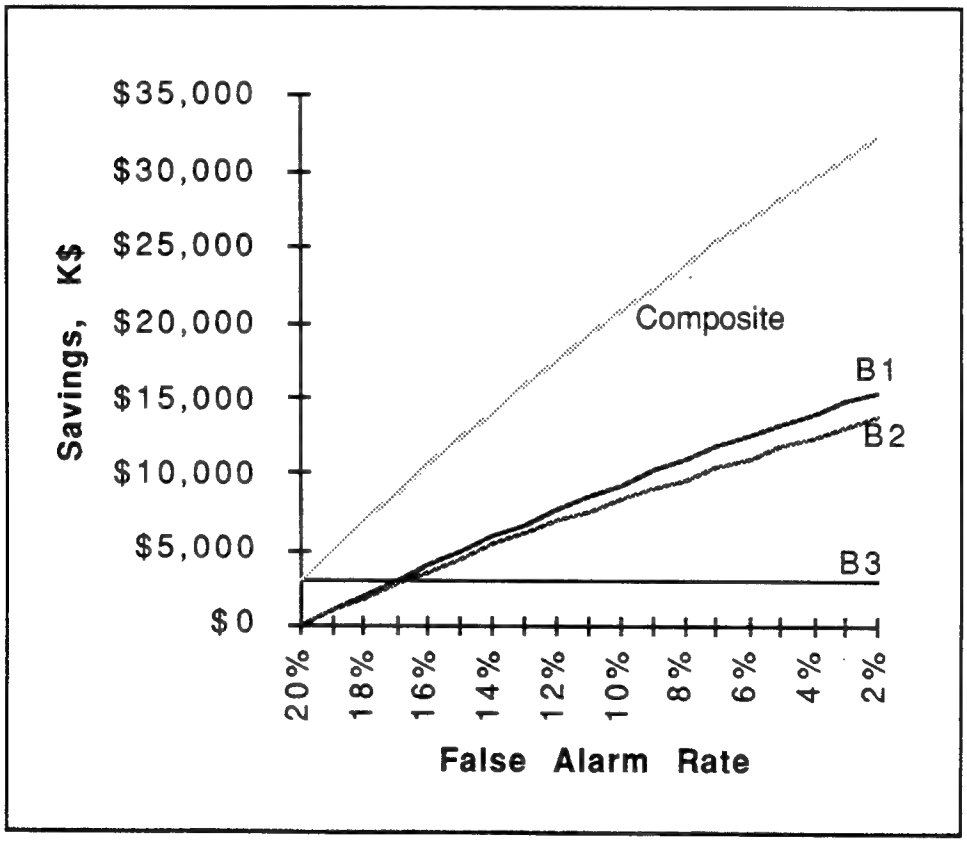


Figure 7.2.2-1. Example Benefits (Savings) Curves

7.2.2.1 Benefit Model Inputs

Table 7.2.2.1-1 lists the specific benefit items input to the spreadsheet and provides a brief explanation of each item.

Table 7.2.2.1-1. Summary of Inputs to Benefit Worksheet

Input Parameter	Description
Cost per Repair Action	Average cost ($K) per unnecessary repair action. Includes cost of shipping module to depot, cost of testing/verifying module and returning to spares.
Average Cost per Abort	Average direct cost ($K) of aborting a mission. Attributable to fuel, manpower, etc.
Fraction of BIT Alarms Causing Maint Action	Average fraction of BIT alarms which result in a maintenance action. Usually close to 1.0.
Fraction of Alarms Causing Mission Abort	Average fraction of BIT alarms causing a mission abort. Highly system/application dependent.
Other Non-Recurring	Non-recurring cost savings attributable to other factors such a reduced spares, reduced T.O and training, etc.
Other Recurring	Recurring (per system) cost savings associated with other factors such as easier accommodation of other changes in the system.

7.2.2.2 Benefit Model Outputs

Based on the inputs and the model, the benefit spreadsheet output is a set of cost savings for a range of FAR between F_i and $0.1F_i$ (i.e., one order of magnitude) at intervals of $0.1\ F_i$. This set of savings is an input to the Cost/Benefit Display spreadsheet (see Section 7.2.3).

7.2.3 Cost/Benefit Display Spreadsheet

7.2.3.1 Cost/Benefit Output Model

The Cost/Benefit Results spreadsheet uses the inputs from the cost and benefit spreadsheets and additional inputs from the user and plots the cost curve, the benefit curve and a net (benefit minus cost) curve for a range of FAR between F_i and $0.1F_i$. Figure 7.2.3.1-1 shows an example of the resulting graphical output. Note that in this example the cost and benefit curves intersect at two points. At a FAR worse than about 15%, the "cost of admission" is not recovered. At a FAR better than about 1%, the exponentially increasing costs again exceed the (approximately) linear benefit curve. The Net curve shows a broad maximum around 6%, which in this example would be the optimum FAR to achieve.

It is entirely possible that the cost and benefit curves never intersect, and the Net curve remains below zero for all values of FAR. This is indicative of a case for which neural network insertion would be of no benefit, at least from a Life Cycle Cost viewpoint.

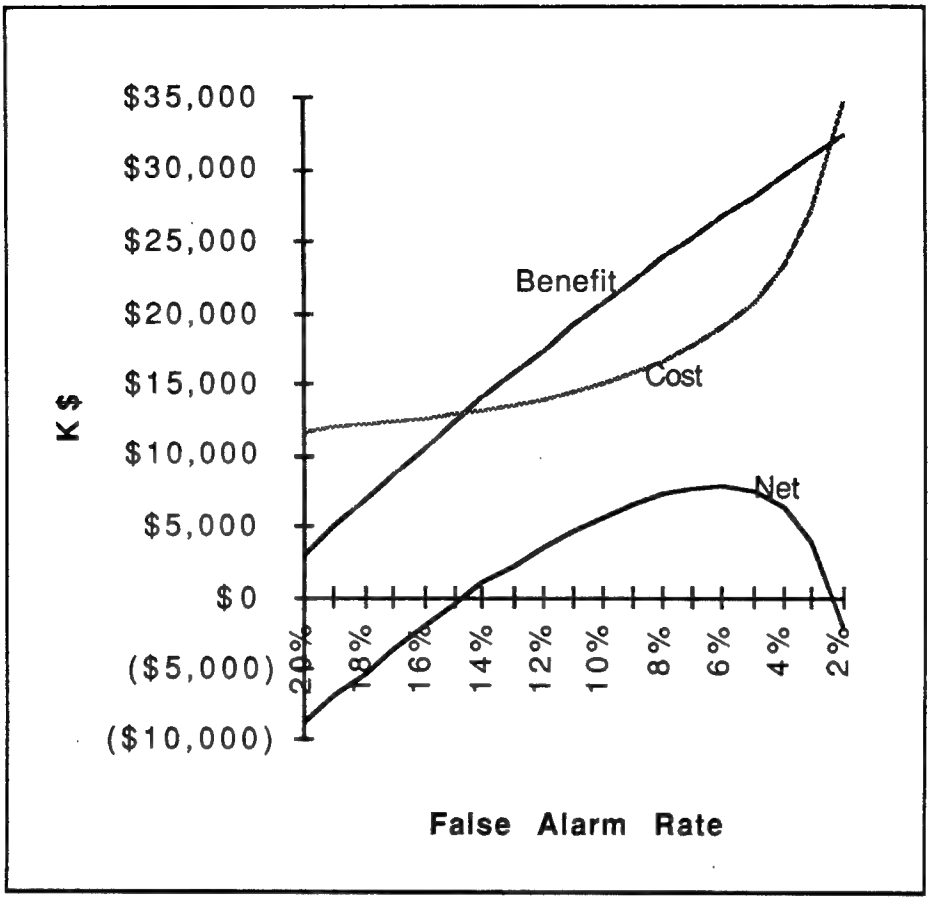


Figure 7.2.3.1-1. Example of Cost/Benefit Results Display

7.2.3.2 Cost/Benefit Display Inputs

Table 7.2.3.2-1 summarizes the user-provided inputs to the Cost/Benefit Display spreadsheet. Parameters needed by other spreadsheets are automatically passed as appropriate.

Table 7.2.3.2-1. Summary of Cost/Benefit Display Inputs

Input Parameter	Description
MTBF	Average MTBF of system
Remaining Years of Service	Expected remaining years during which system will be used and supported
Fraction of BIT Alarms Causing Maint action	Average fraction of BIT alarms which result in a maintenance action. Usually close to 1.0.
Mission Duty Cycle	Average fraction of time that systems are powered up.
Number of Systems	Number of fielded systems
Initial FAR	Fraction representing the initial (baseline) False Alarm rate (F_i)

7.2.3.3 Cost/Benefit Display Outputs

The Cost/Benefit output consists of a display with three major elements. A table showing Total Savings, Costs and Net savings as a function of FAR is displayed. In addition, the same information on a per system basis (including spreading non-recurring costs across the N systems) is displayed. The per system costs are particularly useful to provide perspective to the total cost figures. The third element is a graphical representation of the cost, benefit and net curves plotted as a function of decreasing FAR.

7.3 Results

Using the spreadsheet tool to explore examples, it is possible to draw some general conclusions about the efficacy of neural network insertion at each maturity level. However, it must be emphasized again that there may be numerous exceptions to the general conclusions, and specific systems must be considered on a case-by-case basis.

7.3.1 Mature Systems

In general, mature systems are unlikely to be good candidates for NNFAF technology insertion. They typically are not processor intensive, and lack sufficient reserve processing capacity to implement the requisite processing without expensive upgrades. In addition, because they often were designed to less stringent BIT requirements, they will lack sufficient BIT detectors to allow the NNFAF concept to work effectively. The need to add BIT sensors and processor/memory may cause significant (and expensive) platform impacts in terms of volume, weight cooling and power.

Mature systems also have the least potential for realizing the benefits. The principal problem is that the benefits are "integrated" over a relatively short remaining system life. In addition, the NNFAF approach has the capability to improve over time through collection and analysis of data and periodic retraining. However, the short remaining life does not allow time for NNFAF techniques to yield maximum benefit.

Mature systems may have mitigating factors which would allow a payback from neural network insertion. For example, some older systems have relatively high (e.g., greater than 25%) False Alarm rates, leaving substantial room for improvement on a portion of the cost/FAR curve with a low slope. In addition, a large number of fielded systems and/or a low MTBF would make a mature system a good candidate. Finally, if a mature system is to receive an extensive upgrade which extends its service life, the upgrade may represent a cost effective opportunity to insert NNFAF technology.

7.3.2 Emerging Systems

The benefits of neural network insertion into emerging systems must be evaluated on a case-by-case basis. Many systems acquired within the last ten years were designed and tested to reasonably stringent FAR requirements. (e.g., 3% for Air Force MILSTAR terminals). The costs associated with achieving this performance have already been incurred, and the performance is already good enough that there will be a high "cost of admission" and high costs for incremental improvement. Furthermore, only small improvements in the FAR are possible, since the FAR cannot be better than zero.

Mitigating factors to be considered for emerging systems include a long remaining life, with an associated high likelihood of upgrades on which NNFAF could be "piggybacked"; and a high likelihood of sufficient BIT detector coverage and spare processor and memory capacity. Also, in some cases fielded systems fall short of requirements despite passing all the required maintainability tests. These systems would leave more margin for improvement than the requirements may indicate.

7.3.3 Future Systems

Future systems offer the best possibilities for insertion of NNFAF technology, principally because the system can be designed to accommodate neural networks from the beginning. The "cost of admission " is minimal because it is not necessary to build on an existing system.

It is entirely possible that NNFAF techniques will initially cost more to implement than other more traditional False Alarm reduction techniques. NN filters will likely require more processing and memory than alternate approaches (although processing and memory will continue to become less and less expensive resources), and will also incur the expense of generating initial training data and training the networks. To realize the full effectiveness of NN techniques, it is necessary to collect experiential field data.

7.3.4 MILSTAR Example

As an example of an emerging system, the spreadsheet tools were used to analyze the insertion of NNFAF techniques into the Air Force MILSTAR terminals. Where possible, actual logistics cost information from the Life Cycle Cost data base is used. In other cases, engineering estimates were based on detailed knowledge of the terminal. K1 and K2 are based on engineering judgment.

The MILSTAR terminal system requirement for FAR is 3%, and the maintainability sell-off tests indicate that the terminals will meet this requirement. The estimated "break even" cost to attain the 3% false alarm rate using NNFAF techniques is about $46M, versus a maximum savings at 0% false alarm rate of about $6.2M. Therefore, the benefit curve falls way short of ever crossing the cost curve.

There are two reasons for the lack of payback. The processors in the TAC and BBP are based on a twenty-five year old architecture (the Data General Eclipse) implemented in twenty year old technology (predominantly SSI and MSI integrated circuits). Although there are a few reasonably current microprocessors in the system (e.g., 80286, 8080) and three 2901-based signal processing computers which can be programmed only in a very low level assembly language, no combination of these provides enough throughput to support neural network processing. ROM and RAM memory is also based on older, lower density technology. Therefore the upgrade costs have a high recurring component due to the necessity to totally replace the TAC and BBP processors and memory. This in turn would necessitate re-compiling and re-testing all the application code.

The second reason is that the terminal FA performance is already quite good at 3%. This results in a high incremental cost of improvement with little room for significant gains.

7.4 Impact Study Summary

Initially, Neural Network False Alarm Filtering may cost more to implement than other methods which, at the outset, may work as well. The costs are driven by processing and memory needs, plus the need for generating initial training data and the training of the networks. The major

advantage to NN techniques is the relative ease with which changes may be made as field experience is gained. However, it is unlikely that any one terminal will experience enough failures and false alarms to permit effective retraining of the networks. Furthermore, independent learning and updating would result in a very difficult configuration management problem.

To take full advantage of the retraining capability of NN techniques, it will be necessary to have a system in place to collect and analyze field data across all terminals and periodically distribute revised connectivity and coefficients. Although there are costs associated with the collection and analysis of field data to improve performance, the very systematic nature of neural networks will ultimately lower the cost of improving system performance compared to the traditional "ad hoc" methods usually used, and ultimately result in better performance.

A Global MILSATCOM Maintenance (GMM) system to automate the collection of data and provide central facility analytical tools to help improve both false alarm and module isolation performance is currently being studied under contract to ARPA. (See Figure 7.4-1.)

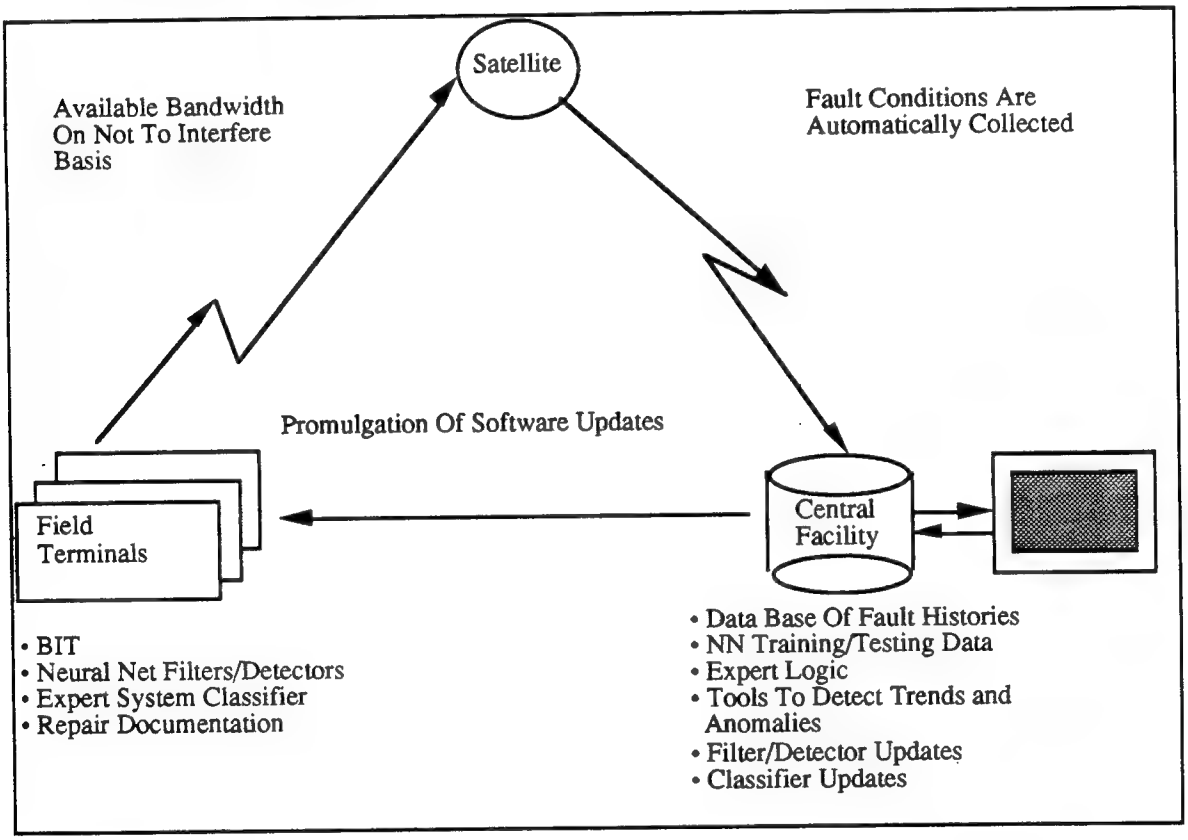


Figure 7.4-1. Global MILSATCOM Maintenance Concept

In the GMM concept, neural networks and expert systems are used in each terminal to provide false alarm filtering and module isolation. Whenever a failure is detected, the terminal retains the associated BIT signature and, via operator/maintainer query, determines the repair action that actually fixed the problem. At the next opportunity, on a low priority not-to-interfere basis, the terminal uses an order wire channel over the satellite to send the BIT signature and repair history to a central facility.

The central facility builds a case record for all the fielded terminals, and provides case-based reasoning tools to assist an analyst in identifying flaws in the expert system logic and the neural networks. Periodically, revised networks are trained and tested and distributed to the terminals via the normal software update paths. The result is that each terminal 'learns" from the experience of all fielded systems, so BIT performance improves over time in a way that is essentially transparent to the operator or maintainer.

In summary, it is difficult to draw any general conclusions regarding the cost-effectiveness of neural network insertion. There are many dependencies, and each case must be considered individually. Therefore the spreadsheets provide a list of factors which affect the costs and benefits of neural network insertion, and a "first order" model to assist in assessing the trade-offs. The following key attributes in combination clearly make a system a good candidate for neural network insertion:

Table 7.4-1. Key Attributes of a Good System Candidate for NNFAF Insertion

• Long Remaining Life
• Large Number of Fielded Systems
• High Present False Alarm Rate
• Low MTBF
• High Cost of a False Pull or Mission Abort
• Ample Reserve Processing/Memory
• Good existing BIT Detector Coverage
• System Already in Place to Support Collection of Field Data
• System Already in Place to Distribute NN Updates

8. DEMONSTRATION

The NNFAF demonstration was the culmination of the NNFAF contract efforts. It was defined in the NNFAF statement of work to be a demonstration of the application of neural networks to improving BIT by filtering out false alarms and identifying intermittent faults. The four demonstration approaches were incorporated into a software demonstration which was conducted at Rome Laboratory on August 24, 1994. This section describes the preparation, conduct, and results of the demonstration.

8.1 Assessment Plan

The deliverable R&D Test and Acceptance Plan (Demo Plan) was used to define the demonstration and evaluation procedures. The demonstration tests consisted of one mini-demonstration test and four demonstration tests. The mini-demonstration illustrated, in a pre-scripted manner, each of the four approaches which were selected during the approach down selection process and implemented using the BIT simulator and the neural network development process. Each approach consisted of a unique neural network model which was implemented to evaluate its performance with respect to a combination of fault report cause and BIT technique. The remainder of the demonstration tests exercised various capabilities of the NNFAF software.

Table 8.1-1 summarizes the demonstration tests.

Table 8.1-1. NNFAF Demonstration Tests

Test	Test Description	Approach
1	Mini-Demo: Demonstrate each approach / network type via pre-scripted visualizations available from NNFAF user interface. Four menu choices correspond to four approaches. Each menu selection demonstrates unique level of BIT fault status reporting.	REINFORCE/Temperature/Viterbi, LRU level SPR/Temperature/Activity Detector, LRU level Backpropagation/G-Load/Parity, Global Fault Table level Backpropagation/Vibration/Parity, Source level
2	Demonstrate performance of SPR network applied to Activity Detector BIT, with Temperature fault report cause (uses mini-demo SPR menu selection).	SPR/Temperature/Activity Detector, LRU level
3	Demonstrate performance of REINFORCE network applied to Viterbi Decoder BIT, with Temperature fault report cause (uses mini-demo REINFORCE menu selection).	REINFORCE/Temperature/Viterbi, LRU level
4	Demonstrate basic functionality of NNFAF system: GUI, use of simulator to generate network training/testing data, use of NNFAF data processing utilities, use of NWorks to build, train, test network file. Demonstrate performance of Backpropagation network applied to Parity BIT, with Vibration fault report cause, using generated network and data.	Backpropagation/Vibration/Parity, Source level
5	Demonstrate use of simulator to generate "noisy" data. Demonstrate performance of Backpropagation network applied to Parity BIT, with G-Load fault report cause, using existing network with generated noisy data.	Backpropagation/G-Load/Parity, Source level

8.2 Evaluation Methodology

The NNFAF software was installed on the Rome Laboratory delivery platform (Sun) immediately prior to acceptance testing. The evaluation of NNFAF was conducted at Rome Laboratory on the Rome Laboratory delivery platform using the R&D Test and Acceptance Plan. All of the demonstration tests were conducted by a Contractor representative (the NNFAF neural network expert) in the presence of a Government representative (the NNFAF Program Manager).

The mini-demonstration was also presented as part of the final NNFAF oral presentation at Rome Laboratory. Each mini-demonstration selection was run. For each, the final neural network architecture and neural network configuration were shown and explained. The related fault report cause and BIT technique were explained. Each network was presented with a set of test data

previously generated by the NNFAF BIT simulator, simulating the effect of the fault report cause on the related BIT technique. The networks were run against the data (tested) and the results of the test pass were displayed via network instrumentation and/or written to a results file.

8.3 Demonstration Results

All of the demonstration tests were successfully conducted in accordance with the R&D Test and Acceptance Plan. Documentation of acceptance (signatures of contractor and Government representative) can be found in the R&D Test and Acceptance Plan in the section "Documentation of Acceptance."

9. CONCLUSIONS

The NNFAF contract addressed the feasibility of using neural networks to filter BIT false alarms and to identify BIT intermittent failures. The work consisted of a state-of-the-art assessment of neural network and BIT technologies; a methodical down selection of four candidate approaches (consisting of a neural network model, a BIT technique, and a fault report cause) for demonstration; software and neural network development efforts to implement a BIT simulator and to develop, train, test and evaluate the neural networks; a study of the impact of neural network false alarm filter insertion into mature, current, and future communication systems, and a demonstration of the approaches.

The results of this effort indicate that neural networks show a potential to filter BIT false alarms and to identify BIT intermittent and hard failures. The neural networks were evaluated using simulated BIT fault report signatures (collections of simulated BIT device status reports over defined intervals of time). Some of the data contained simulated noise, in order to evaluate the robustness or generalization capabilities of the networks.

The results show that not all networks were equally suited to the false alarm filtering task: the Backpropagation and REINFORCE network models were superior to the SPR model. Presented with previously unseen data, both the Backpropagation and REINFORCE models were successful at distinguishing between the four fault signature classes. Both models were reasonably successful at identifying the fault signature classes, given low to moderately noisy data; the REINFORCE model was slightly superior to the Backpropagation model. Neither model performed well with very noisy data.

With regard to the impact of inserting neural network false alarm filters in communications systems, it is difficult to draw any broad conclusions regarding the costs and benefits of such insertions without considering specific systems and circumstances. The results of the impact study show that the best candidates for NNFAF technology are future, yet to be designed systems for which neural network filter requirements can be included in the initial system design.

Finally, the results show that there are many challenging issues which still must be resolved. The first is to use real failure data for network training and testing. One of the potential problems with using real data is that the data may not be available, if the BIT system is not operational when the networks are developed. Another unresolved issue relating to the use of real failure data is fault signature cueing (determining where, in a stream of BIT status reports, the fault signatures begin and end). One of the major advantages of NNFAF technology is the ease of updating through network retraining. To take maximum advantage of this adaptability, a system must be in place to collect field failure data, retrain the networks and distribute revisions to the systems. Several such iterations may be necessary in order to optimize the network performance.

10. LESSONS LEARNED

The major lessons learned as a result of this effort were:

The structured down selection process was a good method of analyzing a large space of potential solutions, given a restricted budget.

The Backpropagation network model performed surprisingly well with the temporal data in this problem domain.

The REINFORCE network also performed extremely well and should be studied further.

The cost/benefit spreadsheets used for the impact study evolved into a beneficial estimating tool.

11. RECOMMENDATIONS FOR FUTURE WORK

Future work for BIT neural network false alarm filtering should address the unresolved issue of training and testing the networks with real failure data. In addition, a logical extension of the BIT neural network false alarm filter is to investigate the use of neural network and other AI technologies to improve fault isolation: typically, incorrect or ambiguous module isolation is also a significant contributor to false pulls. A Global MILSATCOM Maintenance (GMM) system is currently being studied under contract to ARPA. Its concept is to automate the collection of historical system performance data and to provide centrally-located intelligent analytical tools to help reduce false alarms and to improve module isolation performance (See Figure 7.4-1.). Other recommendations for future work involve (1) further study of both Backpropagation and REINFORCE networks in this problem domain, (2) a comparison of neural network filtering techniques with other methods of false alarm mitigation, (3) extending and refining the cost/benefit spreadsheet analysis method used in the impact study to provide a more general-purpose tool for analyzing the impact of technology insertion into communications systems, and (4) an investigation of the approaches to scaling up from one network to multiple networks in a full BIT system,

12. APPENDIX A. BIT BIBLIOGRAPHY AND LITERATURE ABSTRACTS

This appendix contains the BIT bibliography and literature abstracts. The BIT bibliography and literature abstracts were compiled and written during the state of the art assessment task and updated throughout the contract. Note that an asterisk preceding the reference number indicates that the reference was considered especially relevant to the NNFAF effort.

*[1] Blumberg, J. F. "Learning by Examples with Uncertainty for the Adaptive Diagnostic System", Fifth Annual AI Systems in Government Conference, 1990.

A learning technique which applies Cramer's V to a Chi-square statistical value can be applied to improve BIT isolation. This technique was developed under the Navy Integrated Diagnostic Support System (IDSS). BIT fault signatures may look the same for two or more faults. This technique uses feedback from the maintenance personnel to adaptively learn the probability of which fault caused the signature. The feedback is merely the fault isolation report, the isolated faulty unit, and if the isolation report was correct.

[2]Cavanaugh, K. F. "CITS Expert Parameter System (CEPS)", IEEE NAECON, 1987.

The CEPS project is an AI/expert system which was applied to the B1-B to improve maintenance diagnostics. It uses a rules based knowledge expert system which describes rules for determining the health of each function and the SRUs involved with each function. The result is that if a system function fails, then each SRU will be identified with a certainty factor describing the likelihood that the SRU caused the function to fail.

*[3] Cooper, C., Haller, K. A., Zourides, V. G., Skeberdis, P., and Gibson, W. Smart BIT/TSMD Integration, Final Report, RL-TR-91-353.

Report generated by Grumman for Rome Lab on integrating SMART BIT and TSMD (previous Rome Lab studies). Grumman used a simulator to demonstrate the results of implementing SMART BIT and TSMD in a system. Several artificial intelligence (AI) techniques were employed from previous SMART BIT projects. Neural networks and k-nearest neighbor AI techniques were introduced in the SMART BIT system during this project. The TSMD (time stress management device) collects environmental data to correlate to BIT reports and possibly filter out false alarms due to the environment.

The implementation of TSMD data into a SMART BIT system is a good idea. However, the correlation between TSMD data and false alarms does not have known signatures. This project focused on defining how environmental data affects false alarms (FAs) and showing that the system could recognize this. The system should be able to recognize the correlation between FAs and the environment. But, since a model of FA/environment doesn't realistically exist, the system must be able to determine if a correlation between the environment and FAs exists or not. If a correlation does exist, then the system must be adaptable enough to recognize correlations between FAs and the environment that differ from the training models.

[4] Counil, C., and Cambon, G. "Functional Approach to Built-in Selftest of Integrated Digital Filters", Electronic Letters, v. 27 n. 25, 2326-2327, December 5, 1991.

Linear feedback shift registers (LFSRs) are applied to digital filters inputs and outputs for BIT. The input LFSRs generate pseudorandom data, whereas the output LFSR generates a signature.

[5] Davis, K. "Diagnostic Expert System for the B1B", IEEE AES Magazine, April 1988.

The CEPS (Central Integrated Test System (CITS) Expert Parameter System) was developed to improve failure isolation in the B1-B. The CEPS system is described, as well as a discussion of the causes of diagnostic problems.

[6] Gleason, D. "Analysis of Built-in Test Accuracy", 1982 IEEE Reliability and Maintainability Symposium.

A Markov analysis is presented in relationship to BIT detection, isolation, and false alarms. The math behind the analysis is described. However, this analysis is not accurate when applied to systems which have a relatively low failure rate and a low false alarm rate.

[7] Haller, K. A., Zbytniewski, J. D., and Anderson, K. "Smart Built-in Test (BIT)", IEEE AUTOTESTCON, 1985.

SMART BIT is a Rome Lab. project which uses BIT techniques and artificial intelligence to prevent false alarms and intermittent failures. SMART BIT used four basic filtering techniques - 1) Integrated BIT, 2) Information-Enhanced BIT, 3) Improved Decision BIT, 4) and Maintenance History BIT. These techniques are described and were demonstrated in a LISP demonstration system.

[8] Harris, C. M. and Crede, C. E., Shock and Vibration Handbook: Basic Theory and Measurements, Vol. 1, McGraw-Hill, 1961.

The vibration curve was defined based on equations from this book.

[9] Howell, V. J., and McArthur, T. S. "Central Integrated Test System (CITS) Expert Parameter System (CEPS): A Logistics Perspective", IEEE NAECON, May 1989.

CEPS is a large expert system that was applied to the B1-B aircraft to shorten maintenance time and reduce cannot duplicates (CNDs) and retest OKs (RTOKs). A brief high level description of the CEPS program is presented.

*[10] Malcolm, J. G. "Practical Application of Bayes' Formulas", 1983 IEEE Reliability and Maintainability Symposium.

Bayes' formulas can be applied to the area of BIT to show the relationships between fault parameters, system reliability, and failure detection. Fault parameters include retest OK, false alarms, missed faults, and cannot duplicate. The theory and presentation is very good. However, a definition of false alarm was improperly used which resulted in an incorrect observation.

[11] Marvel, D., and Hurst, G. "T-1 Maintenance Takes a New Path", Telephony, June 15, 1992.

Bit error rate can be tested remotely to determine fault location using key physical parameters. The application of this maintenance philosophy is in the telecommunications industry and does not have any practical use in military systems.

[12] Petrov., P. G. "Method for Testable Design and for Built-in Test", CSI/IEEE Symposium on VLSI Design, 1991.

The paper discusses the application of m-class algebra to a design for testability and BIT. The material is not clearly presented. As a result, no benefit can be drawn from this report.

[13] Richards, D. W., and Collins, J. A. "Intelligent Built-in Test and Stress Measurement", IEEE AUTOTESTCON, 1989.

This paper presents the use of both SMART BIT and TSMD to improve future BIT effectiveness and confidence. An introduction to BIT problems and a two level maintenance approach are described. Overviews of the Rome Lab. SMART BIT and TSMD projects are presented as well as the status of both projects.

[14] RADC-TR-81-220, Analysis of Built-in Test False Alarm Conditions, Hughes Aircraft Co., Feb. 1981.

The determination of reports to have been caused by false alarms is very simplistic. Back-up information is not presented for charts or conclusions.

[15] RADC-TR-90-31, A Contractor Program Manager's Testability/Diagnostics Guide, Giordano Assoc., April 1990.

Not a lot of applicable information but some interesting statistics. For example, F-16 had > 13,600 man hours for processing unnecessary removals over a 6 month period. Also, DoD task force (86) found 20-50% maintenance actions resulted in NEOF.

[16] RL-TR-92-82, Design Definition Phase for Micro Time Stress Measurement Device (TSMD) Development, Final Technical Report, Westinghouse Electronic Corp., May 1992.

[17] Rosenthal, D., and Wadell, B. C. "Predicting and Eliminating Built-in Test False Alarms", IEEE Transactions on Reliability, v39, n 4, October 1990.

The effects of BITE measurement noise and bias on BIT reports are examined. Guidelines for BIT measurement limits are presented that can be applied to tests to predict false alarm potential. Probability theory for BIT is also presented.

[18] Shao, J., and Lamberson, L. R. "Markov Model for k-Out-of-n:G Systems With Built-in-Test", Microelectronics and Reliability, v 31 n 1, 123-131, 1991.

A Markov Model for k-out-of-n:G systems is applied to BIT with a detailed presentation of the probability theory. The author feels that a Markov process is the best way to analyze BIT parameters. System reliability functions are generated and presented in several numerical examples.

*[19] Shen, Y.-N., and Lombardi, F. "Concurrent Built-in Self-Test with Reduced Fault Latency", 1991 IEEE International Workshop on Defect and Fault Tolerance on VLSI Systems.

Several new concurrent BIT techniques are presented for equivalent combinational circuits or D-latches in a design. The equivalent combinational circuits are tested by adding BIT circuitry which

monitors the inputs of each equivalent circuit. When the inputs of two agree, then the outputs are verified to agree. If the outputs are not the same, then a failure exists. Both a technique using a ring counter and parity tree are presented for D-latch BIT. Both methods can test out memory cells with low area overhead. These types of BIT designs can easily be implemented in large scale integrated devices. Fault detection latency is addressed and determined to be low for both the equivalent combinational circuit BIT and D-latch BIT. The math behind the fault latency calculations is presented as well as several examples.

[20] Turino, J. "You Can Obtain Boundary Scan's Benefits Despite Use of Some Nonscan ICs", EDN, November 12, 1992.

The basic testability advantages of implementing boundary scan devices in a design are presented. Testing of non-boundary scan components by using adjacent boundary scan devices as "virtual test points" is included.

*[21] Zbytniewski, J. and Anderson, K. "Smart BIT-2: Adding Intelligence to Built-in-Test", IEEE NAECON, 1989.

The SMART BIT 2 is a continuation of the SMART BIT project sponsored by Rome Lab. The goals of SMART BIT 2 are to continue developing effective and efficient methods of improving BIT accuracy and dependability. Several approaches to adaptive BIT/neural networks and temporal models are briefly described. The SMART BIT techniques were demonstrated and the results summarized.

[22] Zourides, V. G. "Smart Built-in-Test (BIT): An Overview", IEEE AUTOTESTCON, 1989.

A description of the history of the SMART BIT project is described. The reasons why SMART BIT was initiated is mentioned. An overview of each phase in the SMART BIT program is presented - SMART BIT, SMART BIT 2, and Integration of SMART BIT for JSTARS. Also, a very brief description of each SMART BIT technique is presented.

13. APPENDIX B. BIT DEFINITIONS OF TERMS

This appendix contains definitions of terms relevant to the BIT domain. Some of these definitions are taken from military or other standards, and some were defined by the NNFAF BIT experts for the NNFAF contract. The source of the definition is indicated following the definition.

Active Built-in-Test (BIT). A type of BIT which is temporarily disruptive to the prime system operation through the injection of test stimuli into the system. (MIL-STD-1309C)

Active Redundancy. That condition where parallel back-up items are operating simultaneously, rather than being switched on when needed. (MIL-STD-1309C)

Algorithmically Generated Pattern. An array of digital data automatically generated by a predetermined software routine or program. The pattern may be generated and applied in real time. (MIL-STD-1309C)

Ambiguity Group. The group of maintenance replaceable units which may have faults resulting in the same fault signature. (MIL-STD-1309C)

Availability. A measure of the degree to which an item is in operable and committable state at the start of a mission, when the mission is called for at an unknown (random) point in time. (MIL-STD-1309C)

Built-in-Test (BIT). An integral capability of the mission equipment which provides an on-board, automated test capability to detect, diagnose, or isolate system failures. The fault detection and, possibly, isolation capability is used for periodic or continuous monitoring of a system's operational health, and for observation and, possibly, diagnosis as a prelude to maintenance action. (MIL-STD-1309C)

BIT, Continuous. A type of BIT which continually monitors system operation for errors. Examples include parity and other error detecting codes. (MIL-STD-1309C)

BIT Fault Signature. The BIT report values (usually Pass/Fail) over a period of time that can be used to classify a failure in the area tested by the BIT routine. (NNFAF)

BIT, Initiated. A type of BIT which is executed only after the occurrence of an external event such as an action by an operator. (MIL-STD-1309C)

BIT, Passive. A type of BIT which is non-disruptive and non-interfering to the prime system. (MIL-STD-1309C)

BIT, Periodic. A type of BIT which is initiated at some frequency. An example is BIT software executing during planned processor idle time. (MIL-STD-1309C)

Cannot Duplicate (CND). A fault indicated by BIT or other monitoring circuitry which cannot be confirmed at the next level of maintenance. (MIL-STD-1309C)

Central Integrated Test System. An on-line test system which processes, records, or displays at a central location, information gathered by test point data sensors at more than one remotely located equipment or system under test (also called system integrated test system). (MIL-STD-1309C)

Collaborative Data. Data that can validate a fault report. For example, a system level failure may validate a lower level fault report. (NNFAF)

Correlation Data. Data that can be used as part of a BIT fault signature.(NNFAF). Some examples are:

- Other BIT reports.
- System state, such as antenna transmitting, special processes occurring, etc.
- Environmental data such as temperature, vibration, G force, and state of other platform equipment.

Equivalent Faults. Two or more faults which create the same response for all possible tests. (MIL-STD-1309C)

Failure. (1) The state of inability of an item to perform its required function. Failure is the functional manifestation of a fault. (MIL-STD-1309C)

(2) Change in operating characteristics of an item resulting in degradation of useful performance. (MIL-STD-1309D)

(3) (Working definition) The inability of an item to perform its specified function, which results in the inability of the system to perform its specified function. (NNFAF)

Failure, Intermittent. (1) A failure which occurs randomly in time. (MIL-STD-1309C)

(2) Failure for a limited period of time, followed by the item's recovery of its ability to perform within specified limits without any remedial action. (MIL-STD-1309D)

(3) (Working definition) A fault which results in a failure that occurs repeatedly over time. Intermittent failures can occur at regular intervals or randomly. (NNFAF) Rules and qualifiers for the intermittent failure definition were also defined:

The system does not meet its requirements when an intermittent failure exists.

If the system is in an over-spec condition then a BIT failure report is invalid and is not an intermittent.

An intermittent failure must be reported at least to the maintenance operator so that a repair decision can be made.

Intermittent failures may not need to be reported during the platform operation depending on the severity of the failure.

Failure Report Latency. The amount of time that it takes to report a failure after the failure occurs. (NNFAF)

False Alarm. (1) A fault indicated by BIT or other monitoring circuitry where no fault exists. (MIL-STD-1309C)

(2) A fault indicated by BIT or other monitoring circuitry where no fault exists. Note that a BIT false alarm is actually a malfunction of the BIT. (MIL-STD-1309D)

(3) (Working definition) An incorrect report to the operator that a maintenance or repair action is necessary when no action should be taken. (NNFAF). Rules and qualifiers for the false alarm definition were also defined:

If a BIT test collaborative signal is OK, then the portion of the system that the signal collaborates is OK.

If the system is in an over-spec condition then a BIT failure report is invalid and is not a false alarm.

If a fault exists but the system still meets its requirements, then a system failure does not exist.

Events that cause false alarms never require a repair action to be taken and should not be reported to the operator during platform operation.

Fault. (1) A physical condition that causes a device, component, or element to fail to perform in a required manner; for example, a short-circuit or a broken wire. (MIL-STD-1309C)

(2) A physical condition that causes a device, a component, or an element to fail to perform in a required manner; for example, a short circuit or a broken wire or an intermittent connection; or, a degradation in performance due to detuning, maladjustment, misalignment, or failure of parts; or, immediate cause of failure (e.g., maladjustment, misalignment, defect, etc.). (MIL-STD-1309D)

(3) (Working definition) The inability of a component to perform the specified requirements of that component. Note that a fault may not affect the specified system operation and as a result is not considered a failure. (NNFAF)

Fault Equivalence. Two or more faults which create the same response for all possible tests. (MIL-STD-1309D)

Fault Isolation. Isolating the fault internal to an integrated component or device such as an integrated circuit. (MIL-STD-1309C)

Fault Latency Time. The extent or duration of time during which the existence of a fault is not known; or, the elapsed time between fault occurrence and fault indication. (MIL-STD-1309C)

Fault, Out-of-Tolerance. A defect or malfunction in a component, assembly or system in which a performance parameter approximates but fails outside the prescribed upper or lower limit for the parameter. (MIL-STD-1309C)

Global Fault Signature. The BIT report values (BIT fault signature) with system operational status (system fault signature) and environmental information over a period of time that can be used to classify a failure. The environmental information includes any information or state that can be gathered which is generated externally from the system. (NNFAF)

Historical Time. Time that spans a relatively large period and includes multiple signatures. (NNFAF)

Local Time. Time within the period of a specific signature. Local time is a subset of historical time. (NNFAF)

Parity Bit. An additional bit used in digital data transmission or memory to make the number of "1" bits it contains either odd or even as appropriate for a given application. (MIL-STD-1309C)

Periodic Check A test or series of tests performed at designated intervals to determine if all elements of the UUT or test system are operating within their designated limits. (MIL-STD-1309C)

Prognostics. The use of test, performance, or other related data in the evaluation of a system or equipment for determining the potential of impending faults. (MIL-STD-1309D)

Pseudo-random Patterns. A repeatable sequence of digital patterns that has the appearance of being random. (MIL-STD-1309D)

Re-test OK (RTOK). (1) The subsequent passing of a previously failed test. (MIL-STD-1309C)

(2) A unit that was identified as malfunctioning in a particular manner at one maintenance level, but that specific malfunction could not be duplicated at a higher maintenance level facility. (MIL-STD-1309D)

System Fault Signature. The BIT report values (BIT fault signature) with system operational status over a period of time that can be used to classify a failure. The system operational status includes any information or state that can be gathered which is generated from within the system. (NNFAF)

14. APPENDIX C. NEURAL NETWORK BIBLIOGRAPHY AND LITERATURE ABSTRACTS

This appendix contains the neural network bibliography and literature abstracts. The neural network bibliography and literature abstracts were compiled and written during the state of the art assessment task and updated throughout the contract. Some of the entries in the bibliography were abstracted and some were not. Those that were not are included here in the subsection entitled Other References.

[1] Adams, Dennis J., Stuart Clary, Yoh-Han Pao, Thomas L. Hemminger, and Percy P. C. Yip. "An Efficient Hierarchical Neural Network Architecture for Discriminating Time-Varying Underwater Signals." Proceedings of the Government Neural Network Applications Workshop - Volume 2, August 1992, 7-11.

The work of this paper focuses on using high-order neural networks to classify underwater signals which are temporally complex. Self-organization techniques are used to determine prototype selection. Multiple functional- link networks with random weight enhancements classify events. The authors describe an iterative approach, using three increasingly complex systems. They also present their method for mapping the temporally complex signals into static "snapshots" which are easily recognizable by neural nets. The final system was a hybrid architecture using the functional link net. The authors present lessons learned for system development time (training the net vs. determining the system architecture) as well as recommendations for scalability, adaptability, and computational efficiency.

RELEVANCE: This is a different approach to the problem of temporality. The lessons learned and recommendations are valuable.

[2] Anderson, Charles W. "Learning to Control an Inverted Pendulum Using Neural Networks." IEEE Control Systems Magazine, April 1989, 31-37.

This paper presents a discussion of how to solve the control problem of the inverted pendulum task by using neural networks. Specifically, the goal is to learn to balance the pendulum with no a priori knowledge of the dynamics: performance feedback is not available at each step and appears as a failure signal. The issues are delayed performance evaluation, learning under uncertainty, learning non-linear functions, and the credit assignment problem. The type of neural network learning methods which are evaluated are reinforcement and temporal difference.

This paper answers the question "How can control be accomplished when neither a model of the system dynamics nor an objective function describing the system's behavior is available?"

The inverted pendulum problem is described, as well as how it has been solved previously using neural networks. This research uses two unsupervised learning networks. The first is the action network which maps the current state into control actions. It is a reinforcement network which is single layer and stochastic and its output is the probability of generating a left or right push control action. The second is the evaluation network, which maps the current state into an evaluation of that state. It is a temporal-difference network (see Sutton). The system was evaluated against Michie and Chambers' BOXES system as a single layer network system with one single binary-valued input. Then it was built with a hidden layer in each network, using backpropagation as the learning mechanism. The system learned indirect non-linear mapping.

RELEVANCE: This paper discusses use of the temporal difference and reinforcement models and application of the temporal difference model using a multi-layer network and backpropagation. The system demonstrates the utility of separating system functions into two networks and linking them together.

[3] Atkins, Robert G. Private Communication.

This communication concerned classification of radar targets from high range resolution profiles using multi-layer perceptron neural networks. The backpropagation learning rule was used. The author experimented with network size vs. real vs. ideal input data (features), as well as with additive noise and alignment uncertainty. The work compared the computational and storage requirements of a neural network backpropagation classifier to profile matching, Euclidean distance, and Mahalanobis distance classifiers. The work concludes that the neural network performs better than conventional classifiers, although it does not take into account the size of the training data sets, nor the length of training time.

RELEVANCE: The paper presented a learned heuristic: it is necessary to match the complexity of the classifier with the complexity of the distribution of the feature vectors. Also, the network must be trained to learn about noise and other feature variations in order to become insensitive to them. Be careful of misleading training.

[4] Barto, Andrew G. and P. Anandan. "Pattern-Recognizing Stochastic Learning Automata." IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-15, No. 3, May/June 1985, 360-375.

This paper discusses the Associative Reward-Penalty (AR-P) algorithm and proves a form of optimal performance. Simulation results are presented which illustrate the task performance and allow it to be compared with other algorithms.

The theory of learning automata is briefly presented, followed by supervised learning pattern classification (note that this paper is PRE-backpropagation). The associated learning method is a minimization of the mean-square approximation error. Then, associative reinforcement learning is described, wherein the learning system and its environment interact in a closed loop. At time step k, the system evaluates itself via the environment, and responds with an action. The algorithm is presented and a convergence theorem is proved. Simulation results are given and a comparison is made with the selective bootstrap algorithm (Widrow et. al.)

RELEVANCE: This algorithm intersects supervised and unsupervised learning methods.

[5] Barto, Andrew G. and Michael I. Jordan. "Gradient Following without Backpropagation in Layered Networks." IEEE First International Conference on Neural Networks, Vol. 2, 629- 636.

This is a method of solving nonlinear supervised learning tasks using multi-layer feedforward networks but without backpropagation. Instead it uses the Associative Reward-Penalty (AR-P) algorithm. It introduces the S-model AR-P variant for learning with real-valued reinforcement. It also discusses optimization techniques such as "batching" for increasing the learning efficiency of AR-P. This research compares the AR-P variants with each other, as well as very briefly with the classical backpropagation method.

A different method for gradient following (or gradient estimating) was developed: a measure of global performance (a scalar "reinforcement" or "payoff" signal) is broadcast to all hidden units in

the network. Each unit must evaluate itself by correlating its activity with the signal and then estimating the partial derivative of this measure with respect to its weights. All interacting units simultaneously vary their outputs to obtain an estimate of the appropriate derivatives. The gradient with respect to weights is computed from these estimates. (NOTE: weights are NOT directly varied).

AR-P networks can perform both supervised learning tasks and associative reinforcement learning tasks. This paper discusses only non-recurrent networks, specifically feedforward. The learning method is mathematically described and the theory is briefly discussed. Optimization was accomplished by allowing the weight updating sequence to take place several times during the presentation of a single input pattern. These weight changes are gathered and added to the actual weights at the end of the presentation. This is called the "batched" method. Also, the direction of steps and the magnitude of the steps in weight space were investigated, since these contribute to the convergence speed.

A benefit of this method is in simpler computational requirements (no recursive error propagation). Also, it may be easier to implement in hardware. A drawback is longer learning time (larger training data sets), especially when the network has multiple output units.

RELEVANCE: This work highlights experimental nature of AR-P model and describes disadvantages. It is related to work by Klopf, stochastic learning automata, Williams' REINFORCE algorithm, and supervised learning.

[6] Bray, Michael A. "Alarm Filtering and Presentation."

This paper discusses alarm filtering and presentation in the control rooms of nuclear power and other process control plants. It is a survey of previous approaches, current research and implementations, new directions in applying AI technology to the problems, and recommendations for the future.

RELEVANCE: It was initially thought that the topic of Alarm Filtering would be relevant. However, this subject refers to filtering the alarms which are presented to a nuclear power plant operator, so that he does not experience information overload, or undue stress in responding to too many alarms. All of the alarms are considered real. None are considered false. There is no reference to neural networks.

[7] Caudill, Maureen. "GRNN and Bear It." AI Expert, May 1993, 28-33.

This article is an introduction to the General Regression Neural Network (GRNN) of D. Specht.

RELEVANCE: Questionable.

[8] Caudill, Maureen. "Neural Networks Primer Part VII (Drive Reinforcement Theory)." AI Expert, May 1989, 51-58.

This article presents a brief description of H. Klopf's Drive Reinforcement Theory, based on classical conditioning, when a problem involves an "omen" signal that serves as a warning of a forthcoming situation to which the network must respond. This model learns the relationship between the omen and the response. This is an unsupervised learning method based on variations of Hebbian learning to incorporate some temporality by looking at changes in signal level.

This currently is a classical conditioning research tool, not used in applications. It has been simulated in software by Gluck, Parker and Reifsnider using frequency-coded simulation (pulse-coded analog signals).

RELEVANCE: Temporality is an issue which is discussed: the system computes the change in the network after each time tick. The accumulation of changes over time determines the level of learning.

[9] Caudill, Maureen. "Neural Network Training Tips and Techniques." AI Expert, January 1991, 56-61.

This article presents tips based on experience and rules of thumb to optimize backprop networks. All tips are based on software model implementation. Briefly, the tips are: how to improve problems with a bad initial set of weights, how to nudge the network into convergence, how to avoid size problems in network scaling, how to work with the momentum term, how to include noise to promote insensitivity to it, and how to experiment with the size of the hidden layer.

RELEVANCE: Methods for optimization and heuristics on network architecture definition are provided.

[10] Chow, Mo-Yuen and Sui Oi Yee. "An Adaptive Backpropagation through Time Training Algorithm for a Neural Controller." IEEE International Symposium on Intelligent Control, August 1991, 170-175.

This work investigates training neural networks to perform control actions for a given cost function and looks at a real-time application.

The authors propose a variation of BPTT which adjusts the number of time steps K as it trains the controller (whereas in standard BPTT, the number of steps is fixed throughout). The result is an adaptive BPTT method which reduces the number of time steps and finds the minimal number required to successfully train the network.

RELEVANCE: Temporality is discussed, as well as a method of optimizing a relatively experimental model which is known to require large memory in order to iterate through the time steps.

[11] Chow, Mo-Yuen and Sui Oi Yee. "Methodology for On-Line Incipient Fault Detection in Single-Phase Squirrel-Cage Induction Motors using Artificial Neural Networks," IEEE Transactions on Energy Conversion, Vol. 6, No. 3, September 1991, 536-545.

This article is similar in content to "Application of Neural Networks to Incipient Fault Detection in Induction Motors" abstracted elsewhere. It describes using a three-layer backpropagation neural network for incipient fault detection in rotating machines, specifically detection of the turn-to-turn insulation faults and bearing wear in an induction motor.

It also describes the development of another neural network to filter out false alarms (the transient state measurements of the motor) while retaining the steady-state measurements, by using a modified competitive learning algorithm. Major beliefs expressed are: (1) neural networks have the ability to learn a desired mapping based solely on examples without having to know the exact mathematical relationship of input to output. (2) Mathematical modeling can be avoided with this approach. (3) The fuzzy logic of fault interpretation is implicitly embedded in the neural network.

(4) Parallel processing allows an increase of inputs without a corresponding increase in computation time. (5) Increasing the number of inputs increases the robustness of the network with respect to measurement noise. (6) The internal structure of the network can be easily changed by short retraining.

The experiments were performed based on the assumption that the motor is in a steady-state condition. A computer simulation of the motor was used to generate training and testing data.

A two-network architecture is presented: the inputs (phase current and rotor speed) enter the Disturbance and Noise Filter Neural Network, and the outputs of this network (steady-state phase current and steady-state rotor speed) enter the Fault Detector Neural Network. The result is the condition of the stator winding and bearing.

The specifics of each network are described, and the simulation results are summarized. The Fault Detector Neural Network was tested separately from the Disturbance and Noise Filter Neural Network, before the two were used in conjunction. The authors conclude that for all the performed test cases, using filtered measurements always yielded the correct diagnosis, while unfiltered measurements often produced false alarms. They indicate that the two-network system could be used in real-time applications.

RELEVANCE: There is direct applicability of neural networks to fault detection and false alarm filtering.

[12] Chow, Mo-Yuen and Sui Oi Yee. "Using Neural Networks to Detect Incipient Faults in Induction Motors," Journal of Neural Network Computing, Winter 1991, 26-32.

This article describes using a three-layer backpropagation neural network for incipient fault detection in rotating machines, specifically detection of the turn-to-turn insulation faults and bearing wear in an induction motor.

Two networks were looked at. The first was a conventional, three-layer (2, 10, 2) backpropagation network, where the two inputs were Phase Current and Rotor Speed. For the second, it was shown that each of the classifications could be partitioned into three regions, whose boundaries could be approximated by quadratic functions of Phase Current and Rotor Speed. The input was expanded from 2 dimensions to 5, forming the high-order network (5, 10, 2).

The fault symptoms were found in the parameters and state values of the motor (bearing wear by mechanical damping coefficient, motor winding insulation fault by change in winding inductance). Measurements were taken directly from the motor. The network was trained using this data which contained different fault conditions. No fuzzy fault interpretation was required using the neural network. No motor modeling was necessary.

The article compared high-order networks with conventional networks and with conventional statistical discriminant analysis. Pattern and batch weight update schemes were also compared in terms of their learning rates.

The results indicated that the high-order network yielded the best performance for both winding and bearing conditions, and that the pattern update training algorithm learns faster than the batch update method. In addition, the high-order network was shown to be a more accurate fault detector than conventional statistical methods, according to the results shown in this article.

RELEVANCE: There is direct applicability of neural networks to fault detection and false alarm filtering.

* [13] Cooper, Charles, Kenneth A. Haller, John D. Zbytniewski, Kenneth Anderson, Jeff Rey Matson. "Smart BIT-2." RADC-TR-89-277, December 1989. (USGO agencies & their contractors, critical technology).

This report continues the work done at Grumman for Smart-BIT. Specifically, 3 new smart BIT techniques are added:

1. Adaptive BIT: This technique recognizes intermittent behavior by identifying where intermittents occurred previously. The user selects the neural network (backpropagation) which trains on known intermittents over a time period and self-adapts, OR the user selects the K-Nearest Neighbor classifier.
2. Temporal Monitoring: This is performed in parallel with Adaptive BIT. It is used to develop a model of faulty behavior, which is a Markov model of intermittent behavior. This technique used Bernoulli random variables to estimate state transition probabilities dynamically. The temporal behavior of faults can be used to predict whether the fault pattern indicates a hard or intermittent fault.
3. Opportunistic Diagnostic BIT: The purpose of this technique is to extract as much diagnostic information as possible out of each failing BIT test. Whereas the Adaptive and Temporal techniques provided detection of intermittents and limited diagnostic information, the Opportunistic Diagnostic is a possible technique for diagnosing the causes of intermittents.

Sixty-four scenarios from 16 faults were selected from FMECA data. They were induced with or without environmental factors as would be measured by a TSMD. There were four failure modes: normal, random, burst, and hard. Scenarios simulated vibration, temperature, and ambient faults.

RELEVANCE: The paper presented implications of a temporal monitoring method coupled with or modeled by a neural network. Also, it discussed how a backpropagation neural network has been used in the false alarm filtering problem.

[14] Corsberg, Dan. "Alarm Filtering: Practical Control Room Upgrade Using Expert System Concepts." InTech, April 1987, 39- 42.

This paper describes the Alarm Filtering System (AFS) implemented by Idaho National Engineering Laboratory (INEL) for nuclear power plants. The system is meant to aid operators in understanding the overall state of the power plant, and to reacting to alarms. It reduces the number of alarms/messages that are presented to the operator. It monitors alarm importance relative to the current system state. It utilizes object-oriented programming and rule bases in an AI enhanced system.

RELEVANCE: It was initially thought that the topic of Alarm Filtering would be relevant. However, this subject refers to filtering the alarms which are presented to a nuclear power plant operator, so that he does not experience information overload, or undue stress in responding to too many alarms. All of the alarms are considered real. None are considered false. There is no reference to neural networks.

[15] Corsberg, Dan. "Effectively Processing and Displaying Alarm Information." IEEE Conference on Human Factors and Power Plants, June, 1988.

This paper describes the Alarm Filtering System (AFS) implemented by Idaho National Engineering Laboratory (INEL) for nuclear power plants. It specifically describes lessons learned

after the system was installed at two facilities, and used by operators. It addresses the problems of alarm display (showing just enough) as well as alarm processing from the operator's viewpoint. The system models the operator's methodology for rapidly analyzing changing alarms. It tunes alarms in and out, based on the current state of the system. The system uses AI techniques to implement the alarm filtering system.

RELEVANCE: It was initially thought that the topic of Alarm Filtering would be relevant. However, this subject refers to filtering the alarms which are presented to a nuclear power plant operator, so that he does not experience information overload, or undue stress in responding to too many alarms. All of the alarms are considered real. None are considered false. There is no reference to neural networks.

[16] Corsberg, Dan and Larry Johnson. "A Nuclear Reactor Alarms Display System Utilizing AI Techniques for Alarm Filtering." ANS Conference on AI and Other Innovative Computer Applications in the Nuclear Industry, September 1987.

This paper describes the Alarm Filtering System (AFS) implemented by Idaho National Engineering Laboratory (INEL) for nuclear power plants. The AFS is a knowledge-based alarm filtering system. It addresses alarm clutter which reduces alarm display effectiveness. Often alarm overload causes operator stress which leads to mistakes. This system is used to identify the most important information and focus the operator's attention on it. Alarm importance is measured relative to the current plant state. The system uses object-oriented programming and rule-bases as AI techniques. It also makes use of a more sophisticated window-based operator display with colors indicating alarm severity. It operates in real-time.

RELEVANCE: It was initially thought that the topic of Alarm Filtering would be relevant. However, this subject refers to filtering the alarms which are presented to a nuclear power plant operator, so that he does not experience information overload, or undue stress in responding to too many alarms. All of the alarms are considered real. None are considered false. There is no reference to neural networks.

[17] Corsberg, Dan and Don Sebo. "A Functional Relationship Based Alarm Processing System for Nuclear Power." International ANS/ENS Topical Meeting on Operability of Nuclear Power Systems in Normal and Adverse Environments, 1986.

This paper describes the Alarm Filtering System (AFS) implemented by Idaho National Engineering Laboratory (INEL) for nuclear power plants. The AFS is a knowledge-based alarm filtering system. It addresses alarm clutter which reduces alarm display effectiveness. Often alarm overload causes operator stress which leads to mistakes. This system is used to identify the most important information and focus the operator's attention on it. Alarm importance is measured relative to the current plant state. The system uses object-oriented programming and rule-bases as AI techniques. It merges functional and structural process models. It also makes use of a more sophisticated window-based operator display with colors indicating alarm severity. It operates in real-time.

RELEVANCE: It was initially thought that the topic of Alarm Filtering would be relevant. However, this subject refers to filtering the alarms which are presented to a nuclear power plant operator, so that he does not experience information overload, or undue stress in responding to too many alarms. All of the alarms are considered real. None are considered false. There is no reference to neural networks.

[18] DARPA Neural Network Study. AFCEA International Press, Fairfax, VA, November 1988.

This is a definitive study and information gathering effort, and is a centralized repository of information. Models which were experimental then have become relatively mature at this writing.

RELEVANCE: This study discusses real-world applicability of the more well-known models. This work provides a basis of comparison with those that are newer and more experimental.

[19] de Vries, Bert, Ronald Sverdlove, Scott Markel, John Pearson and S. Y. Kung. "Classification of Long Signal Segments." Proceedings of the Government Neural Network Applications Workshop - Volume 1, August 1992, 95-99.

This paper presents the Gamma Neural Network, a design which offers a flexible temporal representation for modeling long delays in classifying signals. It uses a gamma memory, which is a combination of the leaky integrator and the tapped delay line. In addition, there were approximately 1000 classes of signals, so the authors addressed this problem as well. They recommended a separate network for each class. This would relieve the unlearning which would occur if one network had to learn all 1000 classes. The authors also recommended a decision-based training strategy similar to perceptron learning.

RELEVANCE: This is a different approach to the problem of temporality.

[20] Dominic, S., R. Das, D. Whitley and C. Anderson. "Genetic Reinforcement Learning for Neural Networks." IJCNN (IEEE), Vol. II, July 1991, 71-76.

This paper describes the use of a genetic algorithm to train a neural network to control an inverted pendulum. This application is suitable to problems where gradient information is not available, or where supervised learning is inappropriate. The genetic algorithm generates the candidate network; the network is evaluated by applying it to the problem and measuring its time to failure. This method is compared to the Adaptive Heuristic Critic network defined by Anderson, which uses the Temporal-Differences learning method.
The comparison shows that the Adaptive Heuristic Critic may learn better when just a failure signal is available; however, the genetic algorithm may be more appropriate at avoiding failure but not as good at finding an optimal control strategy for all possible states.

RELEVANCE: Temporality is discussed. This is an innovative use of a genetic algorithm with a neural network.

[21] Fukushima, Kunihiko, Sei Miyake, and Takayuki Ito. "Neocognitron: A Neural Network Model fo a Mechanism of Visual Pattern Recognition." IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-13, No. 5, September/October 1983, 826-834.

Neocognitron is presented as a mechanism of visual pattern recognition via a PDP-11/34 computer simulation. It recognizes handwritten Arabic numerals. Its strengths are purported to be that it can recognize these same numerals in the presence of considerable distortion in shape. The network is multilayered. It is characterized by cells at deeper layers responding more and more selectively to stimulus patterns, thereby reducing the effects of shift or distortion. The model discussed in this paper utilizes supervised learning.

The paper describes the different types of cells (S, C, V), the layering of cells, and presents the numerical representations of the cell outputs. The process of pattern recognition is described. The

method of supervised training is discussed and mathematically presented. Each layer is also discussed in detail. The recommendation is made that the number of cell planes of each layer should be changed adaptively depending upon the set of patterns which is to be learned.

It was stated by this paper that the computation and memory storage can be reduced in computer simulation by removing references to connections which are not utilized and by generally thinning-out the network, although thinning-out may result in training difficulty, or presentation of more patterns to accomplish the same amount of training.

RELEVANCE: This model is historically important.

* [22] Haller, Kenneth A., Kenneth Anderson, John D. Zbytniewski, Laura Bagnall. "Smart BIT." RADC-TR-85-148, August 1985. (USGO & their contractors, critical technology).

The purposes of this research were to review current available BIT techniques; investigate AI techniques applicable to smart BIT; analyze each potentially useful concept to determine feasibility with special emphasis on minimizing false alarms and identifying intermittents; assess risks and choice of concepts for demonstration; validate selected smart BIT techniques; and document results.

The conclusions of the project were:

1. Smart BIT techniques can improve initial fault type classification and reduce ambiguity groups.

2. Smart BIT techniques can combine AI with existing avionics hardware (without adding test points or fault coverage) to make a smart BIT system with reduced false alarm rate.

3. An AI program with smart BIT must use a hybrid system with multiple AI techniques.

4. Intermittent faults were able to be recognized and monitored using parallel processes and tracking of time-dependent behavior.

5. The use of module simulation helped in software development without HIL.

6. A fixed set of faults could be diagnosed using rule-invoked procedures from a remote location. A central depot could thereby fault isolate for electronic units at field sites.

7. System factors added to decision criteria resulted in improved fault isolation and reduced false alarms.

8. New tools were investigated for future smart BIT applications.

The recommendations of the project were:

1. Diagnostic expert systems need more research (AI for maintenance, ATE).

2. More AI work needs to be done for feedback and state-dependent behavior applications.

3. There is a need for an integrated AI-based design and development tool to produce a declarative description of system and then synthesize a system prototype from this description.

4. There is a need for "knowledge crystallization" - programs to convert rule-based knowledge into a more compact representation runnable on avionics hardware.

5. Micro-miniaturize large capacity LISP processors to allow use in avionics equipment.

6. There is a need for more user-friendly LISP machines.

7. Temporal reasoning (use of monitors to track behavior over time) needs additional research.

8. There is a need for an integrated approach to reliability and maintenance (BIT plus maintenance plus ATE).

9. A more complex technology needs to be analyzed.

10. The technology should be incorporated into field systems when ready.

RELEVANCE: This is the first volume of the smart-BIT studies, which investigate applying AI to BIT.

[23] Hayes, Paul V., R. Timothy Rue, and Samir I. Sayegh. "A Neural Network Approach to Fault Isolation." Proceedings of the Government Neural Network Applications Workshop - Volume 1, August 1992, 25-30.

This paper discusses an approach to using the backpropagation model for classifying (identifying, isolating) faults in a simple digital circuit. The paper first addresses the state of the art of Test Program Set (TPS) diagnostic approaches and surveys the literature available on using neural networks for TPS diagnostics. The paper presents techniques for collecting and preprocessing training data. The paper present simulation results for the simple digital circuit, and plans for the future.

The researchers found that a two-layer network was sufficient to solve their problem, using a linear transfer function. The network size was 54 input nodes, 27 output nodes. Each output node represented a unique fault on the UUT, as well as a "UUT Good" class. The output node with the largest (most positive value) was the classification.

RELEVANCE: There is a good discussion of collecting and preprocessing input data, using stimulus signatures which are a compression of the original stimulus vector. The paper also discusses references to other neural network efforts in the domain, using Kohonen feature maps and the delta rule.

[24] Hinton, Geoffrey E. "Connectionist Learning Procedures." Artificial Intelligence, No. 40, 1989, 184-234.

This is a report which reviews the history and growth of connectionist models and discusses the directions which future work must take to further improve the field. The report looks at simple associative memories, both linear and non-linear. It presents simple supervised learning procedures such as the perceptron and the least squares procedure for binary threshold units. It discusses backpropagation in detail, including applications and variations (reinforcement, iteration, self-supervised). Boltzmann machines are described. Semi-supervised and unsupervised learning

mechanisms are described, including Hebbian learning, competitive learning, reinforcement learning, and genetic algorithm innovations. Deficiencies of each method are presented along the way. The author concludes by pointing out areas for future research.

RELEVANCE: This is a more recent survey of the state of the technology.

[25] Hiotis, Andre. "Insider a Self-Organizing Map." AI Expert, April 1993, 38-43.

This paper provides a high level description of the SOM. It presents some interesting heuristics regarding both SOM and backprop. For example, it recommends setting the number of hidden nodes in a backprop network to the number of features in the input vector. It also states that each node in an SOM is a representative of an input vector. This implies that the size of the SOM will be as large as the number of inputs (or for each new input, a new node is created? This would imply large system growth over time!). The article states that an SOM cannot be overtrained. Training time is usually fast.

RELEVANCE: An unsupervised learning model, with a comparison to backprop.

[26] Hughen, James H. and K. Rex Hollon. "Millimeter Wave Stationary Target Classification Using a High Order Neural Network." SPIE Vol. 1469, 1991, 341-350.

This paper investigated using a High Order neural network (HONN) to classify targets from Ka-band radar return data. The network provided a minimum mean square error estimate of the optimal discriminant but was reported to be easier to train than backpropagation. The authors compared the HONN to Gaussian classifiers and to MLP using backpropagation (via other research results).

Their results did not match their expectations. They found that a second order network did NOT outperform a first order network. Also they related that the advantages of the HONN over a backpropagation MLP would be "ease of training, complexity, and at least as much performance improvement over conventional classifiers."

RELEVANCE: This paper cast a vote for backpropagation for classification.

[27] Kamgar-Parsi, Behrooz and Behzad Kamgar-Parsi. "Clustering with the Hopfield Neural Networks." Proceedings of the Government Neural Network Applications Workshop - Volume 1, August 1992, 129-133.

This paper presents results of experiments with using the Hopfield network (finding the minimum of an energy function) to perform clustering, and compares the results with conventional techniques.

RELEVANCE: This paper is an example of an optimization technique.

[28] Kim, Dae-Young, Sung-Il Chien, and Hyun Son. "Multiclass 3-D Aircraft ID and Orientation Estimation using Multilayer Feedforward Neural Network." IJCNN, Vol. I, 1991, pp. 758-764.

This paper documents the application of backpropagation to identification and orientation estimation of different classes of aircraft in a variety of 3-D orientations. Also, 2-D distortion-invariant feature space was introduced, which describes the aircraft image and is used as input to

the neural network. The work studied the optimum structure (various sizes of hidden nodes and input features) and the reliability of a neural network classifier.

The backpropagation method was partly modified using Fahlman's method for faster convergence.

RELEVANCE: This paper discusses interesting optimization techniques and heuristics.

[29] Klopf, A. Harry. "Drive-Reinforcement Learning: A Real-Time Learning Mechanism for Unsupervised Learning." IEEE First International Conference on Neural Networks, Vol. 2, 1987, 441-445.

This paper presents the Drive-Reinforcement Learning model. It is based on classical conditioning phenomena, a modified Hebbian learning rule, and unsupervised learning. The modification to Hebb is to take into account the passage of time, or sequentiality vs. simultaneity. Neuronal input and output are treated as frequencies, reflecting the frequency of their firing. The network is characterized by primary (fixed) and secondary (acquired or learned) drives. The mathematical model is presented.

RELEVANCE: The real-time implications of modeling classical conditioning phenomena are of interest.

[30] Kosko, Bart. Neural Networks and Fuzzy Systems. Prentice-Hall, Englewood Cliffs, NJ, 1992.

This book is being used as a reference for the Bi-directional Associative Memory (BAM) neural network model which Kosko developed. There are several relevant chapters.

RELEVANCE: BAM is a model which is one of the Neuralworks set.

[31] Leaver, R. A. and P. Mars. "Stochastic Computing and Reinforcement Neural Nets." IEEE ASSP Magazine, April 1987, 4-21.

This paper discusses the Adaptive Reward-Penalty (AR-P) reinforcement learning method. It is unsupervised and replaces the teacher with a critic which provides reward or penalty responses from an environment. The neural network receives these signals and learns to select actions so as to maximize the probability of receiving a reward signal. This method uses stochastic learning automata theory. The signal is a scalar which is used to broadcast a global measure of performance. Each adaptive element (node, unit) correlates its own activity with the reinforcement signal and estimates the partial derivative of the performance index with respect to its own activity. This method was compared with backpropagation: learning speed is slower, but computations are simpler. AR-P did not get caught in local minima; AR-P learning scaled poorly, even when using gradient reinforcement, due to the restrictive nature of the global reward signal. Backpropagation appears to be better suited to solving large problems. The authors' recommendation was to hybridize supervised and reinforcement structures.

RELEVANCE: This paper underlines the usefulness of backpropagation in real-world settings.

[32] Lippman, Richard P. "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, April 1987, 4-22.

This paper provides an introduction to neural networks by discussing six neural network models for pattern classification. These are: Hopfield Nets, Hamming Nets, Adaptive Resonance Theory (ART), Single Layer Perceptron, Multi-Layer Perceptron (with backpropagation learning algorithm), and Kohonen's Self- Organizing Feature Maps. Neural networks are defined. General characteristics of neural network architecture, structure and capabilities are discussed. The areas of potential benefit of neural networks are reviewed. A brief history of the work on neural network models is presented. A comparison between neural networks and traditional classifiers is presented, with block diagrams of the functionality of each. Also, the different tasks which classifiers can perform are delineated (identification, content-addressable memory, vector quantization). A taxonomy is given of six neural networks which can be used for classification of static patterns. The six models are discussed in detail. Algorithms for each model are provided, along with representations of the general network structure, layering and connectivity. Examples of the performance of each type of network are given. Advantages and disadvantages of each are discussed, relative to the others and to classical or traditional classifiers. This paper was written in 1987, when backpropagation was new, and today the six classifiers are some of the more well-known neural network models. Nevertheless, this paper presents a good review and comparison of the six classifiers.

RELEVANCE: This is one of the definitive articles in the literature, with good classifier descriptions and comparisons.

[33] Lippman, Richard P. "Pattern Classification Using Neural Networks." IEEE Communications, November 1989, 47-64.

This paper presents a taxonomy of neural network classifiers, and discusses the following topics in detail: backpropagation, decision tree, how to match classifier complexity to training data, GMDH and High Order networks, K-nearest neighbor, feature-map, learning vector quantizer, hypersphere classifiers, and radial basis function classifiers.

RELEVANCE: This paper contains a good classifier comparison and an excellent list of references.

[34] McAulay, Alastair and Ivan Kadar. "Neural Networks for Adaptive Shape Tracking." SPIE Vol. 1099, 1989, 74-82.

This paper builds a neural network simulation to solve the problem of tracking and maintaining identification of aircraft with similar-looking silhouettes, accounting for translation, rotation, changing scale or changing aspect. Two neural networks are built: backpropagation and split inversion. The split-inversion network computes the weights for both the output and the hidden layers so as to minimize the square error at the output of the network. The two are compared. Backpropagation converged slowly for networks with few nodes, and failed to converge for larger networks. The split-inversion results are provided to 160,000 interconnects, whereas the backpropagation did not converge at this size. The mathematical derivation for the split-inversion network is not provided. In addition, this was a simple problem which did not account for real-time factors.

RELEVANCE: The author states in his conclusion that the network may have application for time sequential pattern recognition or association.

[35] Morgan, James S., Elizabeth C. Patterson, and A. Harry Klopf. "A Network of Two Drive-Reinforcement Neurons that Learns a Solution to a Real-Time Dynamic Control Problem." First Annual INNS Conference, September 1988.

This paper presents an application of the Drive-Reinforcement Learning model to solve the pole-balancing control problem. The model is based on classical conditioning phenomena, a modified Hebbian learning rule, and unsupervised learning. The modification to Hebb is to take into account the passage of time, or sequentiality vs. simultaneity. Neuronal input and output are treated as frequencies, reflecting the frequency of their firing. This article demonstrates the important differences between primary (fixed) drives and secondary (acquired) drives. The effects of varying learning rates on skill acquisition and retention were also studied.

RELEVANCE: The real-time implications of modeling classical conditioning phenomena are of interest.

[36] Piche, Stephen W. and Bernard Widrow. "First-Order Gradient Descent Training of Adaptive Discrete-Time Dynamic Networks." RL-TR-91-62, May 1991.

This paper discusses training recurrent neural networks (also known as discrete-time dynamic systems with adaptive parameters) using first-order gradient descent algorithms. A definition of a standard discrete-time dynamic system is given. The ordered derivative is defined. Epochwise and on-line training are defined. Two types of learning algorithms are described and compared: the first is based upon the discrete Euler-Lagrange equations and the second is based upon a recursive update of the output gradients. Both are also presented with both epochwise and on-line learning capabilities. The epochwise discrete Euler-Lagrange method is shown to be equivalent to BPTT. The on-line recursive method is shown to be equivalent to recursive backpropagation. The on-line versions are shown to be equivalent, and it is also shown that selection of an appropriate gradient descent algorithm can be based solely upon computational and storage requirements. A computational/storage requirements comparison of the two algorithms is given, as well as explanations of how to reduce computational complexity. Examples are provided for a non-linear controller and an adaptive filter. Applications include pattern recognition, nonlinear control, adaptive control and adaptive digital filtering.

RELEVANCE: The characteristics of on-line training in a temporal domain are of interest, as are state-of-the-art variations of backpropagation.

[37] Rogers, Steven K. and Matthew Kabrisky. "1988 AFIT Neural Network Research." Air Force Institute of Technology, 1988.

This paper briefly presents a summary of recent (1988) research at the Air Force Institute of Technology in the area of neural networks. The Air Force investigated the following topics: improving error-driven learning algorithms, speech recognition, target classification, time series prediction, and optical and VLSI implementations. The error-driven learning algorithms were Glass-Mackay nonlinear differential delay equation, an approximation to Newton's method, and a combination of a Kohonen network with a MLP. Also, other elements of the error-driven learning methods were examined, including modification of the error criterion, the sigmoid function, and the basic structure of the processing element.

RELEVANCE: Some optimization techniques were discussed.

[38] Rumelhart, D., D. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation," in D. Rumelhart and F. McClelland, eds., Parallel Distributed Processing, Volume 1, MIT Press, Cambridge, MA, 1986, 318-362.

This is the classic backpropagation reference.

RELEVANCE: This paper also introduces the backpropagation through time model.

[39] Shyne, Scott S. "The Implementation of Neural Nets as Adaptive Pattern Classifiers." Masters Thesis, State University of New York Institute of Technology at Utica/Rome, 1989.

This masters thesis discusses solving the problem of battlefield target recognition by using Adaptive Resonance Theory (ART1) and Kohonen's Self-Organizing Map (SOM) neural network paradigms. These methods are chosen because they lend themselves to unsupervised feature extraction. These neural network models are not the usual choices for solving the ATR problem; however, they performed successfully. The pattern recognition problem is defined; the applicability of neural networks to this particular type of problem is discussed. A brief history of neural networks is given for reference, including SLP, MLP, backpropagation, Hopfield and Hamming networks. ART1 and Kohonen's SOM are discussed in detail. (ART2 is mentioned briefly, but was not used in this work.) For each type of network, a description of the simulation, test, evaluation and recommendations are given.

RELEVANCE: Use of these types of networks to solve the ATR problem represents to some extent a break from the traditional MLP classifier solution, especially in the area of unsupervised feature extraction and feature learning. Therefore, the results of this thesis are interesting.

[40] Sutton, Richard S. "Learning to Predict by the Methods of Temporal Difference." Machine Learning, 3.9.44, 1988, 9-43.

The Temporal Difference (TD) class of incremental learning procedures is introduced and defined. This method consists of using past experience with an incompletely-known system to predict its future behavior. Credit assignment is accomplished by means of the difference between temporally successive predictions. This paper argues that most of the problems to which supervised learning have been applied (i.e., classification) can really be categorized as prediction problems, and that the TD method can be applied to them. The paper describes a brief history of prediction and TD. It compares and contrasts the TD approach to prediction with the supervised-learning approach. Single-step vs. multi-step predictions are distinguished. It is shown that the linear TD (1) procedure produces the same per-sequence weight changes as the Widrow-Hoff learning method, but the TD can be computed incrementally. The TD family of learning procedures are described. Examples are given which demonstrate that the TD method can converge more rapidly and make more accurate predictions than supervised-learning methods. The theoretical foundation of TD is provided. Optimality and learning rate are considered. Ways of extending the TD methods to be used for more realistic (complex) problems are given. A brief comparison between TD and backpropagation focuses on temporal credit assignment vs. structural credit assignment.

RELEVANCE: The paper discusses problems which involve temporal sequences of observations and predictions. Note: This model is more experimental, so investigators must be careful to scale the method up so it can handle complex nonlinearities.

[41] Toomarian, N. "A Concurrent Neural Network Algorithm for the Traveling Salesman Problem." Oak Ridge National Laboratory, Oak Ridge, TN, January 1988.

This paper presents a solution to the Traveling Salesman Problem using an unsupervised feedback learning mechanism based on a continuous synchronous neural network using the Lagrange multiplier method to minimize the objective function. The concurrency was achieved using the NCUBE hypercube multiprocessor.

RELEVANCE: The hardware implementability of the network is of interest.

[42] Tveter, Don. "Getting a Fast Break with Backprop", AI Expert, July 1991, 36-43.

This paper summarizes some methods of improving the performance of the backpropagation learning method. These methods are: (1) periodic vs. continuous weight updates: periodic uses less CPU time but requires more iterations; it may enable use of a larger learning rate, promoting faster convergence; (2) piecewise linear approximation to replace computing the sigmoid activation function; (3) (Fahlman) add a small positive offset to the derivative of the sigmoid, to avoid slowness when the output is close to 1 or 0 and the value of the derivative is very small (so very little learning takes place); (4) (Chen and Mars) use the differential step-size algorithm: drop the derivative of the sigmoid altogether for the output layer, and replace the inner layer learning rate with one which is 0.1 of the learning rate for the outer layer. This is recommended for continuous weight updates; (5) (Jacobs) Delta-Bar-Delta: adjusting the values of the various learning parameters as the network learns. Each weight has its own learning rate which will be changed according to how well the network converges. The article illustrates how to implement this.

The paper contains good pointers to the original papers in the literature. His experiments were done on the XOR problem and an encoder problem. His results show that the differential step-size algorithm is most effective; he had mixed results with the Delta-Bar-Delta. His conclusion is that "there is probably no occasion where using the original method is appropriate" since there are several simple techniques which can improve it.

RELEVANCE: This paper recommended several backpropagation optimization techniques.

[43] Vemuri, V. "Artificial Neural Networks: An Introduction with an Annotated Bibliography." Lawrence Livermore National Laboratory, Livermore, CA, January 27, 1988.

This paper discusses the biological background of neural networks, presents their evolutionary development along with mathematical formalizations, describes how neural networks had been implemented to solve real problems as of that date, and briefly mentions areas of further research.

RELEVANCE: This work contains an extensive annotated bibliography, sorted into a variety of useful categories such as Historical Works, Survey or Review Material, Contemporary Works, and Technologies and Tools. It also contains an excellent reference list.

[44] Yaworsky, Paul S. and James M. Vaccaro. "Neural Networks, Reliability and Data Analysis." RL-TR-93-5. Rome Laboratory, Griffiss Air Force Base, NY, January 1993.

This paper discusses the importance of being able to understand the knowledge contained in data, for Reliability applications. The goal of the authors was to investigate the characteristics of data which could then be used to develop automatic methods of statistical data analysis. They built a Statistical Neural Network whose architecture was tailored to the content of the input data.

RELEVANCE: The work is not directly related to the NNFAF project; however, the discussions about data, information, knowledge (and their relationship to neural networks) are important.

[45] Watrous, Raymond L. "Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization." IEEE First International Conference on Neural Networks, Vol. 2, June 1987, 619-627.

This paper discusses neural network learning as an optimization problem, focussing on iterative deterministic gradient methods. The backpropagation algorithm is considered in the context of nonlinear optimization. In addition, two higher-order optimization methods (Davidon-Fletcher-Powell and Broyden- Fletcher-Goldfarb-Shanno) are considered in light of computational complexity, convergence properties, and suitability to parallel machines. These algorithms iteratively approximate the second derivative of the objective function. The performance of these algorithms is compared to steepest descent and backpropagation. The paper gives a definition of optimality via the optimality criterion, or the objective function (which is generally nonlinear). Different optimization techniques (search vs. gradient) are compared and it is concluded that gradient methods are appropriate were the error surface is smooth and the error is computable, whereas search can avoid small local minima if the solution space is sufficiently small. Also, stochastic vs. deterministic methods are compared, to show why deterministic methods were selected for this study. Methods of steepest descent and quasi-Newton methods are described. The backpropagation algorithm is presented, along with a discussion of the effects of learning rate and momentum terms. Computational complexity is compared across the four algorithms. The paper concludes that first order methods are economical in space but "extravagant" in time. Second order methods achieve rapid convergence near the minima but require extra space. The choice must be made based on the specific problem requirements.

RELEVANCE: This was a comparison of backpropagation and other methods of optimization.

[46] Werbos, Paul J. "Backpropagation Through Time: What It Does and How To Do It." Proceedings of the IEEE, Vol. 78, No. 10, October 1990, 1550-1560.

This paper first reviews basic backpropagation and then presents the basic equations for BPTT. It discusses BPTT applications (pattern recognition in dynamic systems, systems identification, and control). It provides pseudocode as a means of better understanding the algorithms. It briefly discusses (but does not prove) the chain rule for ordered derivatives upon which BP and BPTT are based.

RELEVANCE: This provides an in-depth presentation of the BPTT equations. The pseudocode is helpful but is not guaranteed to be correct!

[47] Williams, Ronald. J. "A Class of Gradient-Estimating Algorithms for Reinforcement Learning in Neural Networks." IEEE First International Conference on Neural Networks, Vol. 2, June 1987, 601-608.

This report introduces a new general class of algorithms for solving associative reinforcement learning problems using neural networks which are made up of stochastic processing units. This algorithm class is called REINFORCE. The algorithms are interesting because they are trained via a scalar reinforcement signal fed back to the entire net, they stochastically follow the gradient of a performance measure in such problems, making them analogous to the backpropagation method of supervised learning, and they have an on-line implementation even for problems of training temporally extended behaviors in recurrent networks (not true for backpropagation). The report

defines associative reinforcement learning and how the network and its environment interact with time-varying data. A distinction is made between this type of learning and supervised learning (instructive vs. evaluative environmental feedback). The benefits of associative reinforcement learning over supervised learning are discussed. The issue of credit assignment is defined in the context of the learning paradigm (combined structural and temporal). Also, definitions of on-line and off-line learning are presented. The mathematical derivation of the REINFORCE class algorithms is given for both non-temporal and temporal behaviors.

RELEVANCE: This paper discusses the topic of training temporally extended behaviors in recurrent networks.

[48] Williams, Ronald. J. "On the Use of Backpropagation in Associative Reinforcement Learning." IEEE Second Annual International Conference on Neural Networks, Vol. 1, July 1988, 263-270.

This paper explains how backpropagation can be used to train networks to perform associative reinforcement learning tasks. One method is to train a second network to model the reinforcement signal from the environment and then backpropagate through this network into the first. The second is to use the REINFORCE algorithm, backpropagate through deterministic parts of the network and perform a correlation computation where the network is stochastic. Another is to extend the previous method to backpropagate through the stochastic parts of the network also. Associative reinforcement learning is defined. It is stated that a supervised learning problem may be given to a deterministic-stochastic mixed network The terms "backpropagation" and "backpropagation learning algorithm" are distinguished. The mathematical notation for the REINFORCE class of algorithms is presented. A discussion is given on how to incorporate backpropagation into a reinforcement learning problem. These types of networks are experimental and more research and analytical investigation is required. The paper explains how the REINFORCE algorithms are related to backpropagation and explains how a REINFORCE algorithm must work in a network containing both deterministic and stochastic units: the REINFORCE computation is made at stochastic units, and backpropagation is used through the deterministic units. The concept of "backpropagating through a random number generator" is presented. The applicability to supervised learning and search properties using continuous multiparameter distributions are discussed.

RELEVANCE: This approach lends itself to the "creation of algorithms able to take advantage of both [stochastic and deterministic] types of information."

[49] Williams, Ronald. J. and Jing Peng. "An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories." Neural Computation 2, 1990, 490-501.

This paper discusses a variation of the backpropagation through time (BPTT) learning algorithm which is intended to be used on continuously running recurrent networks for temporal supervised learning tasks such as sequence classification or sequence production (both inputs and/or outputs may be time-varying). It has been designed specifically for efficient computation. It is called a cross between "epochwise" BPTT and "truncated" BPTT and is abbreviated BPTT(h:h'). Recurrent network learning algorithms are reviewed, specifically BPTT, real-time recurrent learning (RTRL), and recurrent backpropagation. Five important properties of the hybrid algorithm are given: on-line learning, general purpose, designed to train networks to perform arbitrary time-varying behaviors, designed for time-efficient implementation, minimizes computation per time tick, experimentally verified. It is believed to provide the "computational efficiency of BPTT while retaining the on-line character of RTRL". Formal assumptions and

definitions are given for the network architecture (semi-linear) and dynamics (discrete-time). A network performance measure is given for a sequential supervised learning task. The related approaches of epochwise BPTT, real- time BPTT, and truncated BPTT are discussed, and the improved BPTT(h;h') algorithm is presented and compared to epochwise and truncated BPTT. Experimental performance is discussed; results are presented in a similar report. Some implications and benefits of the algorithm are discussed.

RELEVANCE: Learning arbitrary time-varying behaviors is directly applicable.

[50] Williams, Ronald. J. and David Zipser. "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks." Neural Computation, 1, 1989, 270-280.

[51] Williams, Ronald J. and David Zipser. "Experimental Analysis of the Real-Time Recurrent Learning Algorithm." Connection Science, Vol. 1, No. 1, 1989, pp. 87-111.

These papers discuss the real-time recurrent learning (RTRL) algorithm type, for recurrent networks running continually, as a candidate for solving temporal supervised learning tasks. These networks can learn tasks which require retention of information over varying-length time periods. Sóme experiments are described and the results presented and analyzed: XOR, sequence recognition, learning to be a Turing machine, learning to oscillate. NOTE: These algorithms are computationally expensive and require non-local network communication.

RELEVANCE: The problem of temporality is addressed. NOTE: This model is still very experimental.

[52] Zahirniak, D. R. "Probabilistic Classification Using Radial Basis Function Neural Networks." June 26, 1992.

This paper discusses probabilistic classification methods and shows how three can be implemented using neural networks: K-nearest neighbor (KNN), probabilistic neural network (PNN), and radial basis function neural network (RBF). The paper concludes that RBF performs as well as the optimal KNN or PNN networks but is less sensitive to the network parameters. RBF uses the weights connecting hidden layer nodes to the output layer nodes to desensitize the classification performance. RBF is a supervised learning method.

RELEVANCE: RBF is one of the new models which will be available with the next Neuralworks upgrade.

[53] Zipser, David. "A Subgrouping Strategy that Reduces Complexity and Speeds Up Learning in Recurrent Networks." Neural Computation, Vol. 1, No. 4, 1989, 552-558.

A major drawback of recurrent networks is the computation time needed to update the encoding of the historical information (P): in a network with N recurrently-connected units and M external units, there are N P values associated with each weight. This method divides the original network into subnets for error propagation, while leaving it undivided for activity propagation. It reduces the computation time without changing the connectivity.

RELEVANCE: This paper presents a solution which could improve the suitability of recurrent neural networks to solving large-scale real-world problems.

14.1 Other References

[1] Barto, Andrew G. "Adaptive Neural Networks for Learning Control: Some Computational Experiments." IEEE Workshop on Intelligent Control, 1985, 170-175.

[2] Barto, Andrew G. "An Approach to Learning Control Surfaces by Connectionist Systems," in Arbib and Hanson, eds., Vision, Brain and Cooperative Computation. MIT Press, Cambridge, MA, 1987, 665-701.

[3] Barto, Andrew G. "Connectionist Learning for Control," in Miller, Sutton, and Werbos, eds., Neural Networks for Control, MIT Press, Cambridge, MA, 1990.

[4] Barto, Andrew G. "Game Theoretic Cooperativity in Networks of Self-Interested Units." AIP Conference Proceedings 151, 1986, 41-46.

[5] Barto, Andrew G. "Learning by Statistical Cooperation of Self- Interested Neuron-Like Computing Elements." Human Neurobiology, Vol. 4, 1985, 229-256.

[6] Barto, Andrew G. "Neural Problem Solving." COINS Technical Report 83-03, University of Massachusetts at Amherst, October 1981.

[7] Barto, Andrew G. "Some Learning Tasks from a Control Perspective." COINS Technical Report 90-122, University of Massachusetts at Amherst, December 1990.

[8] Barto, Andrew G., Richard S. Sutton, Peter S. Brouwer. "Associative Search Network: A Reinforcement Learning Associative Memory." Biological Cybernetics, Vol. 40, 1981, 201-211.

[9] Barto, Andrew G., Richard S. Sutton. "Landmark Learning: An Illustration of Associative Search." Biological Cybernetics, Vol. 42, 1981, 1-8.

[10] Barto, Andrew G., Charles W. Anderson and Richard S. Sutton. "Synthesis of Nonlinear Control Surfaces by a Layered Associative Search Network." Biological Cybernetics, Vol. 43, 1982, 175-185.

[11] Barto, Andrew G., Richard S. Sutton and Charles W. Anderson. "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems." IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-13, No. 5, September/ October 1983.

[12] Carpenter, Gail A. and William T. Ross. "ART-EMAP: A Neural Network Architecture for Learning and Prediction by Evidence Accumulation."

[13] Carpenter, Gail A. and Stephen Grossberg. "The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network." Computer, March 1988, 77-88.

[14] Carpenter, Gail A. and Stephen Grossberg. "ART2: Self- Organization of Stable Category Recognition Codes for Analog Input Patterns." Applied Optics, December 1987, 4919-4930.

[15] Carpenter, Gail A., Stephen Grossberg, Natalya Markuzon, John H. Reynolds, and David B. Rosen. "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised

Learning of Analog Multidimensional Maps." IEEE Transactions on Neural Networks, Vol. 3, No. 5, September 1992, 698-712.

[16] Gallant, Stephen I. "Connectionist Expert Systems." Communications of the ACM. February 1988, Vol. 31, No. 2, 152- 169.

[17] Grossberg, Stephen. "Adaptive Pattern Classification and Universal Recoding: I. Parallel Development and Coding of Neural Feature Detectors." Biological Cybernetics. Volume 23, 1976, 121-134.

[18] Hecht-Nielsen, Robert. Neurocomputing. Addison-Wesley, Reading, MA, 1988.

[19] Jordan, Michael I. "Attractor Dynamics and Parallelism in a Connectionist Sequential Machine." Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986, 531-546.

[20] Krile, T., S. Rothstein, A. McAulay, B. Juang. "Polynomial Neural Networks for Airborne Applications." NAECON 1989, 682- 685.

[21] McAulay, Alastair D. "Optical Neural Network for Engineering Design." NAECON 1988, 1302-1306.

[22] McAulay, Alastair D. and Jae Chan Oh. "Image Learning Classifier System Using Genetic Algorithms." NAECON 1989, 705- 710.

[23] McAulay, Alastair D. and Ramesh Ravula. "Interpolation Results in the Split Inversion Neural Network Algorithm." NAECON 1989, 695-697.

[24] Mumford, M. L., D. K. Andes, and L. L. Kern. "The Missile-Borne Integrated Neural Network Demonstration Program." Proceedings of the Government Neural Network Applications Workshop - Volume 2, August 1992, 33-37. Target Classification.

[25] Norrod, Forrest E., Michael D. O'Neill, Erann Gat. "Feedback- Induced Sequentiality in Neural Networks."

[26] Pearlmutter, Barak A. "Learning State Space Trajectories in Recurrent Neural Networks." Neural Computation 1, 1989, 263- 269.

[27] Smotroff, Ira G., Timothy P. Howells and Steven Lehar. "Meteorological Classification of Satellite Imagery Using Neural Network Data Fusion." ICJNN 1990, Vol II, 23-28.

[28] Sutton, Richard S. "Learning to Predict by the Methods of Temporal Difference." Machine Learning, 3.9.44, 1988, 9-43.

[29] Sutton, Richard S. "Reinforcement Learning is Direct Adaptive Optimal Control." IEEE Control Systems, April 1992, 19-22.

[30] Sutton, Richard S. and Andrew G. Barto. "Toward a Modern Theory of Adaptive Networks: Expectation and Prediction." Psychological Review, Vol. 88, No. 2, 1981, 135-170.

[31] Tesauro, Gerald. "Practical Issues in Temporal Difference Learning." Machine Learning, 8, 1992, 257-277.

[32] Vogl, T. P., J. K. Mangix, A. K. Rigler, W. T. Zink, and D. L. Alkon. "Accelerating the Convergence of the Back-Propagation Method." Biological Cybernetics, Vol. 59, 1988, 257-263.

[33] Williams, Ronald J. "Gradient-Based Learning Algorithms for Recurrent Connectionist Networks." April 1990.

[34] Williams, Ronald J. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." Machine Learning, 8, 1992, 229-256.

[35] Williams, Ronald J. "Toward a Theory of Reinforcement-Learning Connectionist Systems." NU-CCS-88-3, July 1988.

[36] Williams, Ronald J. and Jing Peng. "Function Optimization using Connectionist Reinforcement Learning Algorithms." Connection Science, Vol. 3, No. 3, 1991, 241-268.

[37] Yee, Richard C., Sharad Saxena, Paul E. Utgoff and Andrew G. Barto. "Explaining Temporal Differences to Create Useful Concepts for Evaluating States." Eighth National Conference on Artificial Intelligence (AAAI), 1990, 882-888.

[38] Yen, Matthew M, Michael R. Blackburn and Hoa G. Nguyen. "Feature Maps Based Weight Vectors for Spatiotemporal Pattern Recognition with Neural Nets." IJCNN 1990, Vol II, 149-155.

15. APPENDIX D. NEURAL NETWORK DEFINITIONS OF TERMS

This appendix contains definitions of terms which are relevant to the neural network domain. These definitions have been compiled from the literature which was surveyed for the state of the art assessment task. In particular, two sources contained lists of definitions which were helpful in selecting which terms should be defined. These are Neural Computing (NeuralWorks Professional II/PLUS and NeuralWorks Explorer) by NeuralWare, Inc., 1991, and the DARPA Neural Network Study, AFCEA International Press, November, 1988.

1. ADALINE (ADaptive LInear NEuron, Widrow, 1959): A member of a family of trainable pattern-classifiers which distinguishes between patterns using linear discriminate functions. It can only classify linearly separable problems. The ADALINE was one of the first attempts to model biological learning.

2. ART-1 (Adaptive Resonance Theory 1, Grossbery, 1976): An unsupervised neural network model used in pattern classification. The network discovers pattern classifications on its own, in real time. It forms categories for input data, with the granularity of the categories determined by a vigilance parameter. The learning method is based on the assumption that inputs which share a greater number of features should fall into the same category. ART-1 was designed to classify binary input patterns.

3. ART-2 (Adaptive Resonance Theory 2, Grossberg and Carpenter, 1987): ART-2 is similar to ART-1 except that it was designed to classify analog inputs.

4. Annealing schedule: See recall schedule.

5. Associative Memory: A memory which allows retrieval of information by presentation of inexact or incomplete stored memory keys. In an associative memory network, the memory units are the connection weights and the values of the weights are the current state of the network knowledge. The network is presented with an input and it must respond with the associated output. If a trained network is presented with a partial input it will choose the closest match in memory to that input, and generate an output which corresponds to a full input. If the network is auto-associative, then presentation of a partial input will result in its completion by the network. These networks are also known as content-addressable memories and the learning method is also called associative learning.

6. Asynchronous mode: In this mode, processing elements fire randomly and independently of others.

7. BPTT (Backpropagation Through Time, Rumelhart, Hinton & Williams, 1986): An extension of the backpropagation learning method which can be used with problems that involve system dynamics over time. The output error is propagated back through the time path in this model. Since the propagation progresses backwards, the model requires a memory of previous time periods.

8. Backpropagation (Parker/Rumelhart, 1985/1986): A learning algorithm for updating weights in a multilayer feedforward network that minimizes mean squared mapping error (error between\ the designed output of the network and the actual output). It distributes the responsibility for the output error across all elements and connections by propagating it backward through the

connections to the previous layer, and repeating until the input layer is reached. This is one of the most utilized network models.

9. BAM (Bidirectional Associative Memory, Kosko, 1987): A hetero-associative version of the Hopfield network.

10. Boltzmann Machine (Ackley, Hinton & Sejnowski, 1985): A supervised learning algorithm in which network states are determined by "simulated annealing." This type of network uses a noise process to find the global minimum of a cost function.

11. CCN (Compound Classifier Network, 1989): This is a network which classifies inputs against templates by a nearest neighbor algorithm. It utilizes dynamic hidden node allocation when an input is not sufficiently similar to an existing template. This model is similar to Nestor's Restricted Coulomb Energy (RCE) model.

12. Cascade Correlation (Fahlman, 1990): This is a type of supervised learning, multilayer network in which new hidden nodes are added one at a time. Its purpose is to predict the current remaining output error in the network and reduce it by creating the new hidden node. First, the new hidden node is correlated with the current network error, and then the new node is added to the network to form a cascade.

13. Competition: A method of neural network learning which involves intra-layer processing element interaction. Only one or a few elements win and therefore produce an output. Lateral inhibition is sometimes used to effect this.

14. Competitive Learning: An unsupervised learning algorithm in which groups of processing elements compete to respond to a set of stimulus input patterns. The winner within each group is the one whose connections make it respond most strongly to the pattern; the winner then adjusts its connections slightly toward the pattern that it won.

15. Connectivity: Types of interconnections between processing elements. A network can be fully, locally, or sparsely connected. Networks may also be layered, and the elements in the layers can be connected by means of feedback or feedforward connections.

16. Convergence: In some networks, after a finite number of presentations of training data, the values of the network's weights approach the set of values which represent the classification which has been provided in the training data.

17. Counterpropagation (Hecht-Nielsen, 1987): This is a nearest- neighbor classifier which selects from a set of exemplars (templates) by allowing them to compete against each other and selecting one winner. The winner is then decoded into a classification.

18. Delta Rule Learning: In this learning method, the goal is to reduce the error between the actual and the expected outputs by modifying incoming connection weights.

19. Energy Function: Each network state has an energy value which is defined by the output of its processing elements. The recall process iteratively modifies the network state so that the system's energy decreases. This results in a state which represents a local minimum in the energy surface.

20. Fan In: The number of processing elements that excite or inhibit a given unit.

21. Fan Out: The number of processing elements directly excited or inhibited by a given unit.

22. Feedback Network: A network in which some of the connections feed backwards through the network. Sometimes feedback is used to create time-sensitivity.

23. Feedforward Network: A network in which all the connections are from lower layers to higher layers, and there are no feedback connections from one layer to another or from one layer to itself. In these networks, information is passed from the input layer, possibly through middle layers, to the output layer. Often this transfer of information involves the use of a transfer function.

24. Generalization: The capability of the network to respond with a reasonable response when it is presented with incomplete, noisy, or previously unseen input.

25. Hamming Network (Lippmann, 1987): This network implements a minimum error pattern classifier for binary vectors, where the error is defined using the Hamming distance. It is also called the unary model.

26. Hebbian Learning: In this learning method, a connection weight is increased if both its input and the desired output are high. This correlates biologically to the activation on each side of a neural synapse.

27. Hidden Units: These are the processing elements of a network which are neither in the input layer nor the output layer. They are located between these two layers (in the hidden layer) and allow the network to perform more complex problem-solving (non- linear mapping).

28. Hopfield Network (Hopfield, 1982): This is a fully-connected, feedback network which uses unsupervised learning as a pattern classifier. It can also be used to solve combinatorial optimization problems such as the Traveling Salesman. It is also called the Cross-bar Associative Network.

29. Kohonen SOM (Self-Organizing Map, Kohonen, 1979-1982): This is an unsupervised model used for optimization and pattern classification. It does not require explicit training of input-output correlations but "spontaneously self-organizes." It is used to visualize topologies and hierarchical structures of higher-dimensional input spaces.

30. LVQ (Learning Vector Quantization, Kohonen, 1988): This network assigns vectors to classes. It uses a Kohonen (SOM) layer to learn and perform the classification.

31. Learning: Adapting connection weights in response to stimuli presented at input (and optimally) at output.

32. Learning Algorithms: These are the equations which modify (some of) the weights of processing elements in response to input signals and values provided by the transfer function. The learning algorithms allow the processing elements' responses to inputs to change (learn) over time. A learning algorithm is also called a learning rule.

33. MADALINE (Multiple ADALINE, Widrow, 1960): This is a network of ADALINES cascaded together so that non-linearly separable problems may be addressed.

34. Multi-layer Perceptron (Rosenblatt): A multi-layer feedforward network that is fully connected and is typically trained by the backpropagation learning algorithm.

35. Neocognitron (Fukushima, 1980): This is a pattern classifier which combines an unsupervised learning algorithm with a multi-layer architecture. It is tolerant of positional shifts, geometric distortion, and scale variation.

36. Normalization: Normalization takes the vector of values corresponding to the output of a complete network layer and scales it so that the total output is some fixed value. The result is that the total activity in the layer remains approximately constant.

37. On-Line Learning: On-line learning is also referred to as adaptive learning or self-supervised learning. The network learns in real time. There is no distinct training period, as there is with supervised learning.

37. PNN (Probabilistic Neural Network, 1988): This is a neural network implementation of the Bayesian classifier statistical method. The PNN uses training data to develop distribution functions that are used to estimate the likelihood of a feature (input) being within a category (class).

38. RTRL (Real-Time Recurrent Learning, Williams & Zipser, 1989): This is a newer, more theoretical model which is recurrent and can deal with time-varying input or output.

39. Recall: How the network processes a stimulus presented at the input and thereafter creates a response at the output.

40. Recall Schedule: A method of controlling recall parameters as the reverberation progresses (in a feedback network).

41. Recirculation (Hinton & McClelland, 1988): This is an alternative to a backpropagation network in which errors are passed backwards through the feedforward connections. In this model, data is processed only in one direction, and connections are both forward and back. It uses the same learning rule as backpropagation; the connections are separated so that a hardware implementation might be less difficult.

42. REINFORCE (Williams, 1987): This is a class of gradient- estimating algorithms for reinforcement learning. It is an on- line learning method which can learn temporal behavior.

43. Reinforcement Learning: An external "teacher" or "critic" indicates by a scalar value (often binary) whether the system's response to an input is "good" or "bad". The expected output is not shown to the system, as is the case in supervised learning. It requires a reinforcement signal as training feedback.

44. Relaxation: This is the idea that computation proceeds by iteratively seeking to satisfy a large number of weak restraints. Connections represent constraints on the co- occurrence of pairs of processing elements. The network settles (relaxes) into a solution rather than calculating it.

45. Reverberation: Information reverberates in the network until some convergence criterion is met; then the information is passed to the output. This occurs in feedback networks.

46. Self-Organization: This is the method of autonomous modification of the network system dynamics via learning in some or all of its connections to achieve a specified result.

47. Self-Supervised Training: A means of training adaptive neural networks. Self-supervision is used by automata which require internal error feedback to perform some specific task.

48. Simulated Annealing: A stochastic computational technique derived from statistical mechanics for finding near globally-minimum cost solutions to large optimization problems.

49. Single-Layer Perceptron (Rosenblatt, 1957): This is a trainable pattern classifier which classifies using linear discriminate functions. It is one of the original neural network models. Its goal was to model the pattern recognition capability of the visual system. It is very similar to the ADALINE. Its major weakness is that it cannot be used for non-linearly separable problems.

50. Spatio-Temporal Pattern Recognition (Hecht-Nielsen, 1986): This model is a classifier used for recognizing sequences of events over time. It can be used for recognizing repetition, for example, repeating signals. Also, the Kurogi model.

51. Supervised Learning: The system is trained to respond to a given input with a corresponding output by showing it the expected output. This is accomplished in several ways; most notably Hebbian learning, Delta Rule learning, or competitive learning (competition). It is also called hetero-associative learning.

52. Synchronous Mode: In this mode, all or selected elements fire simultaneously at each time step.

53. Temporal Difference (Sutton, 1988): A class of incremental learning procedures which are specialized for prediction (using past experience with an incompletely known system to predict its future behavior). This method assigns credit for error by means of the difference between temporally successive predictions. Learning occurs when there is a change in prediction over time.

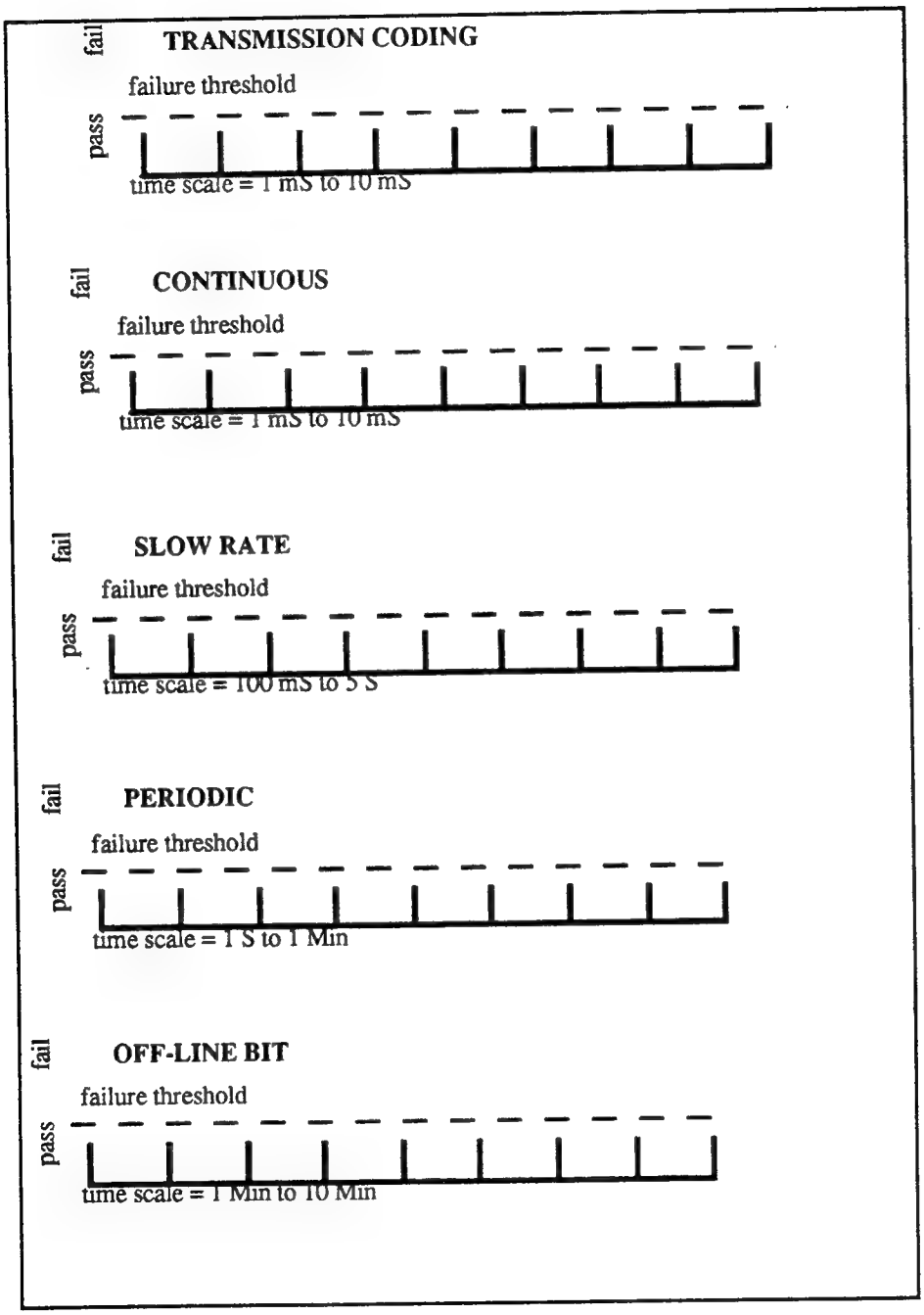
54. Transfer Function: The mathematical function used to control the output of a processing element. Common transfer functions are: hard limit (threshold), which limits the output to either 0 or 1; linear, with which the output can equal the input; and sigmoid, which squashes the input into a range.

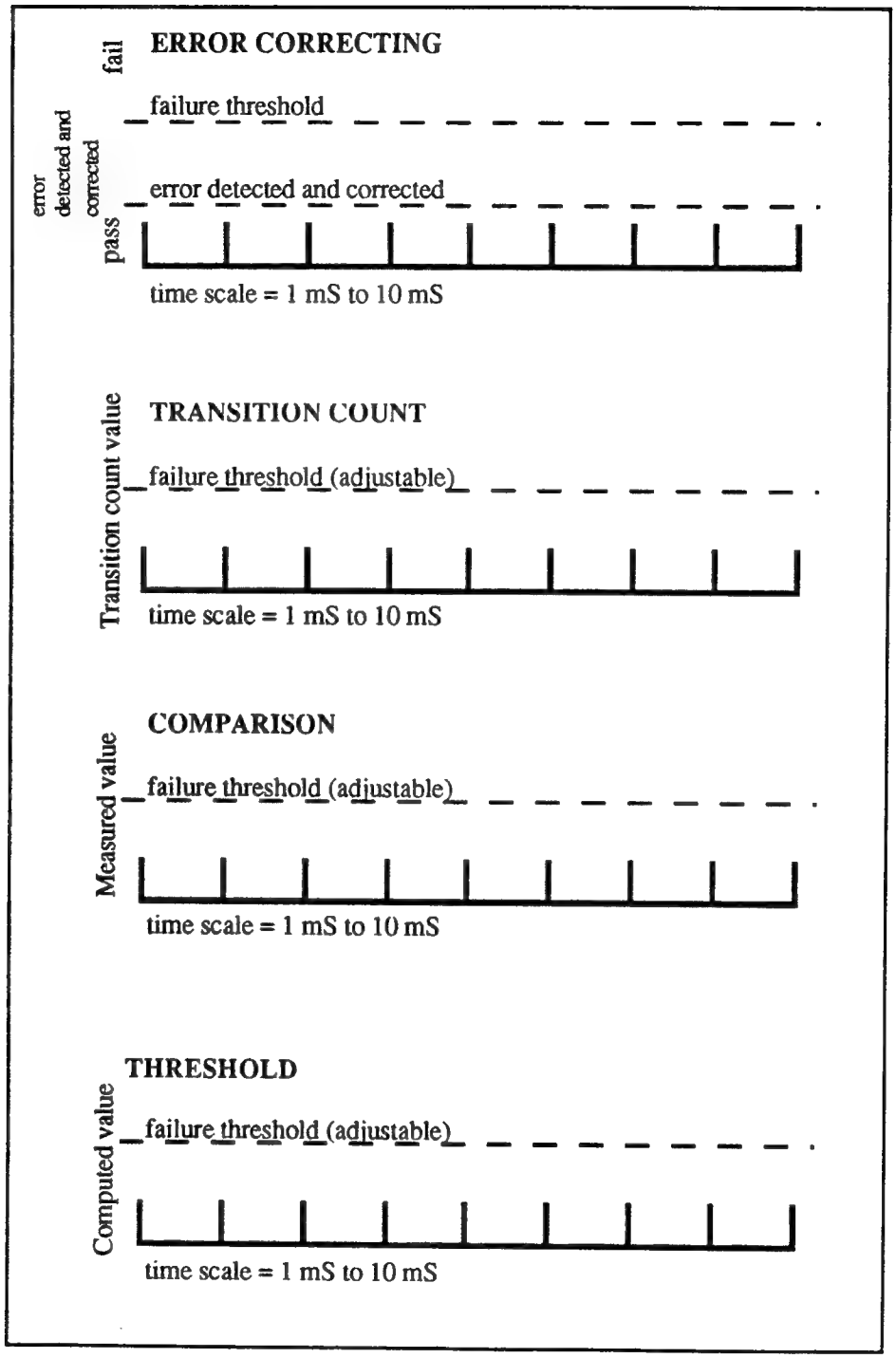
55. Unsupervised learning: The system receives only input stimuli; the network iteratively organizes and reorganizes itself so that each processing element responds strongly to a different set of input stimuli. These response sets represent clusters in the input space which may represent distinct real world concepts.

56. Weight: The value or strength of a neural network processing element's connection to another element. All weights input to an element are combined into a simple value in order to update the processing element's potential.

16. APPENDIX E. NNFAF BIT TECHNIQUE FAULT SIGNATURES

This appendix contains visualizations of the NNFAF BIT Technique fault signatures. The figures show fault report status to the left of each signature and sampling rate below each signature.




ERROR CORRECTING
fail
failure threshold
error detected and corrected
error detected and corrected
pass
time scale = 1 mS to 10 mS
TRANSITION COUNT
Transition count value
failure threshold (adjustable)
time scale = 1 mS to 10 mS
COMPARISON
Measured value
failure threshold (adjustable)
time scale = 1 mS to 10 mS
THRESHOLD
Computed value
failure threshold (adjustable)
time scale = 1 mS to 10 mS

17. APPENDIX F. FAULT REPORT CAUSE CURVE FILES

The default environmental curve files were generated using mathematical functions. This made any curve adjustments, such as scaling and number of sample points, easy to accomplish. It also facilitated analytical analyses of the curves at various points, because each curve is modeled by an equation. However, it is important to understand that the environmental curves are representations of curve shapes that may be possible for various environmental conditions. The most significant characteristics of the three curves are that they have very different shapes. The uniqueness of each shape type will be useful in determining how well the neural networks can perform with each of the types. The curves were represented mathematically because it makes modification and evaluation easier, and not because they are intended to be exact mathematical models of actual environmental conditions.

17.1 Temperature Curve

The temperature curve was designed to be a relatively smooth curve starting at a low magnitude, increasing to a peak, and then decreasing to a low magnitude. The changes in magnitude are gradual because the effect of temperature within an electronic system is usually gradual due to the conduction and convection cooling which exists within typical systems. The curve is represented with a peak at the center because this is easy to model and real data coming from an operational system can easily be formatted such that the peak is at a center point.

The temperature curve function is based on a sine function which was then adjusted using exponents and a line. These various functions were used to take a basic sine shape and make the rate of change in the sine curve more gradual. The resulting functions are:

(1) $Y = [\ (\text{Sine}(x) + 2)^{0.05} - 0.99\ ]\ /\ [\ (3^{0.05} - 0.99) + 0.08x - 0/0.08\ ]$
where $-3.14<x<0$

and

(2) $Y = [\ (\text{Sine}(x) + 2)^{0.05} - 0.99\ ]\ /\ [\ (3^{0.05} - 0.99) + 0.08(1-x) - 0/0.08\ ]$
where $0<x<3.14$.

Equation (1) is mixing the exponentially adjusted sine function with a line of gradually increasing slope. Equation (2) is mixing the exponentially adjusted sine function with the same line, but of gradually decreasing slope. The exponential adjustments and the line both help to smooth the sine curve into a smaller rate of change in curve slope. The resulting temperature curve is shown in Figure F-1.

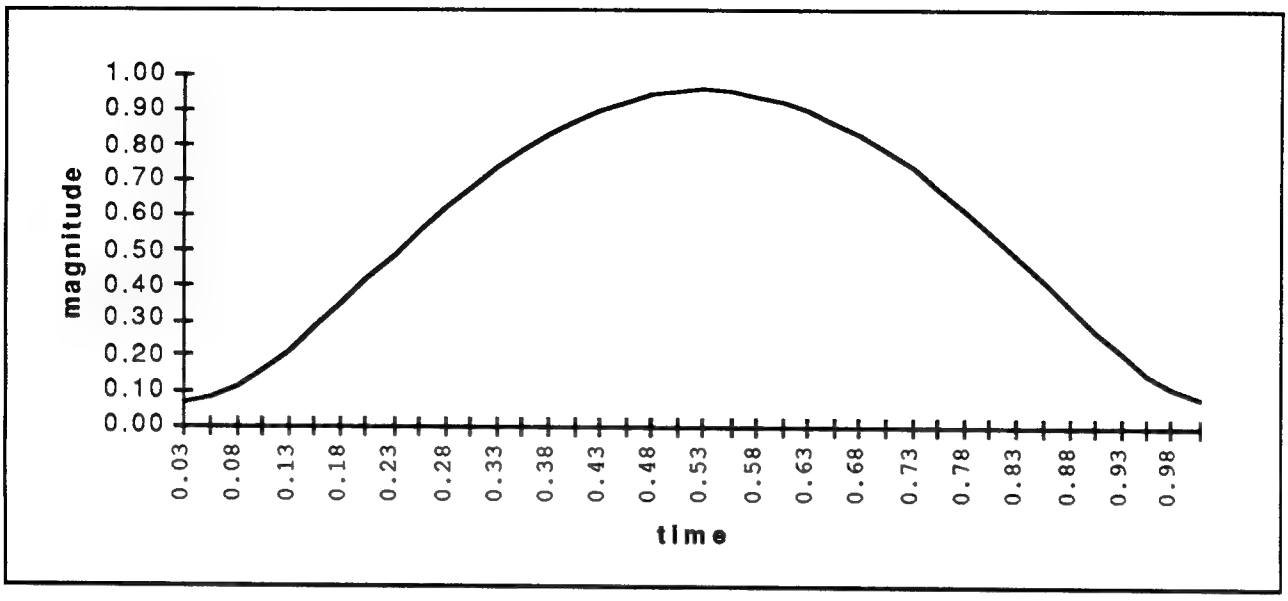


Figure F-1. Default Temperature Curve

17.2 G-Load Curve

The G-Load curve is represented as a low magnitude with a very rapid increase to a peak that stays constant for a period of time and then rapidly decreases to a low magnitude again. This curve is meant to represent the magnitude of G-load when an aircraft executes a maneuver or steep turn, or when an aircraft has hit a large air pocket.

The function used for the G-load curve is:

$$(3)\quad Y = \frac{1}{X^2}$$

where $-2 < X < 2$ and $Y=4$ if $Y>4$.

The range of x was selected to produce the desired shape, or rate of change in the symmetric increase and decrease in the curve magnitude.

The resulting G-load curve is shown in Figure F-2.

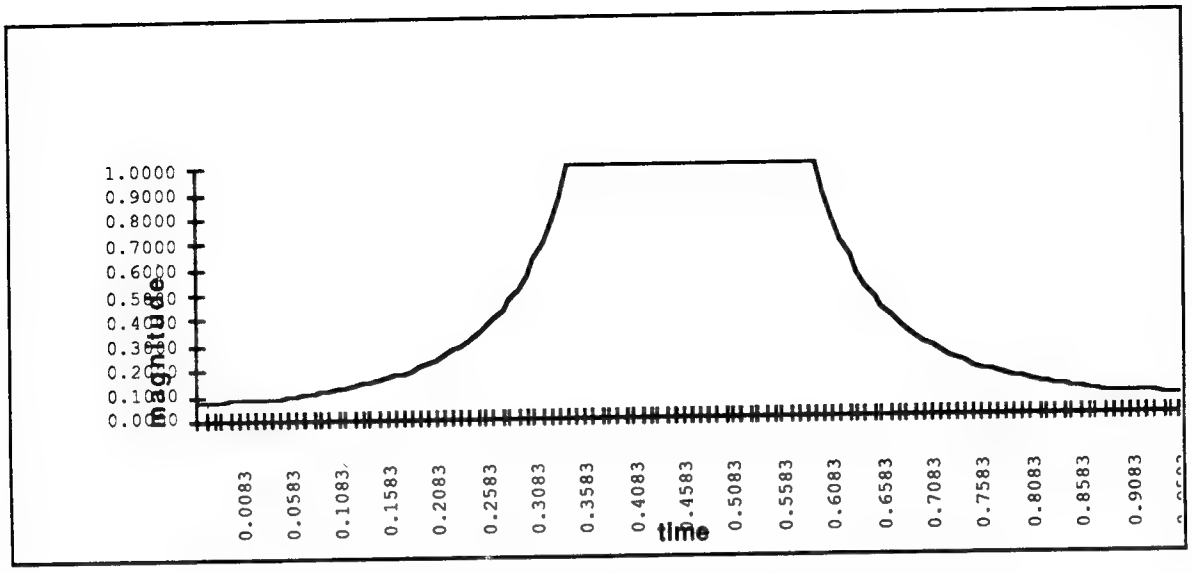


Figure F-2. Default G-Load Curve

17.3 Vibration Curve

Describing a vibration curve in relation to its effect on a system's BIT is very difficult. Many vibration curves exist for various system platform types. Several vibration profiles exist within the same platform depending upon the location of the unit under consideration. These vibration profiles usually consist of magnitude and ranges of frequencies. For an aircraft they may include frequencies in the range from 10 to 2000 Hz. The vibration experienced by a unit can therefore appear to be a realm of random frequencies. However, when these random frequencies are intoduced to a unit, the frequencies with the greatest impact to the unit are the unit's resonant frequencies. The resonant frequencies are natural frequencies inherent in the physical layout of the unit. As a result, even though the frequencies at the platform are for the most part random, natural frequencies will appear at the unit.

The curve was based on the vibration that would be experienced on a typical circuit card (resonant frequencies) within a typical SATCOM system. The resonant frequencies within the card are 140 Hz (first mode), 386 Hz (second mode), and 756 Hz (third mode). More resonant modes exist but they are too small to be included. The resonant frequencies were generated using an Angular Natural Frequency equation [Harris] for a fixed-fixed (built-in) assembly. The magnitude of the resonant frequencies decreases with each higher-order mode, representative of the magnitude decrease for harmonics. The vibration equation is a combination of first, second, and third order resonant frequency modes that have peak magnitudes of 0.7, 0.25, and 0.05 respectively. This curve is shown in Figure F-3.

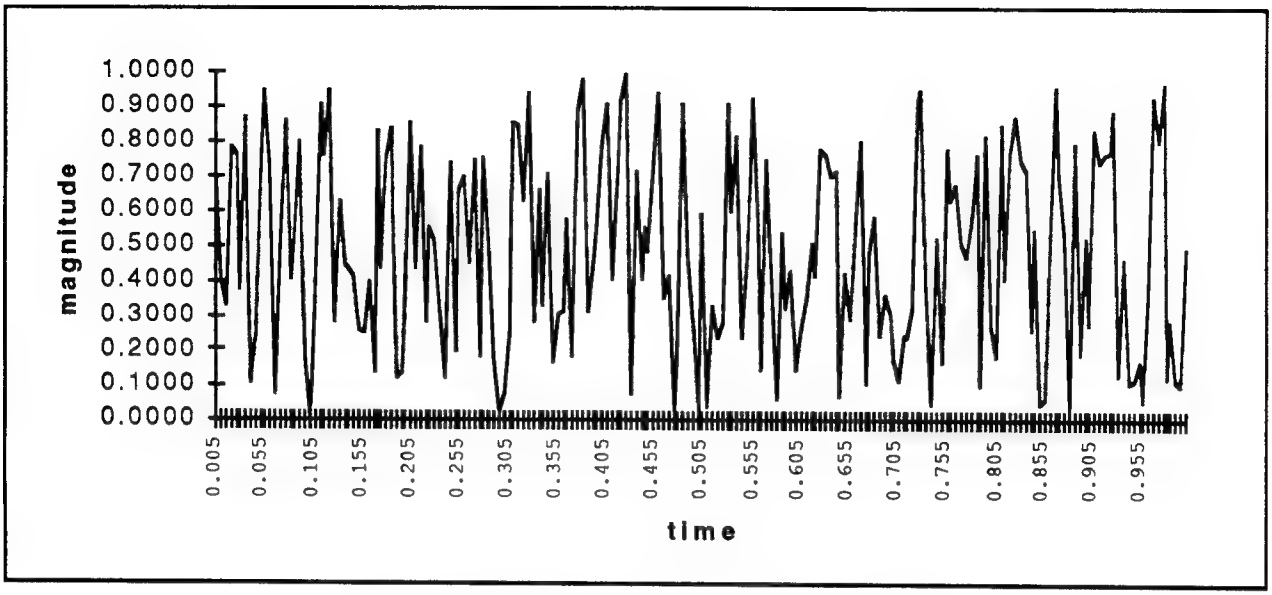


Figure F-3. Default Vibration Curve

18. APPENDIX G. EXAMPLE FAULT SIGNATURE DATA FILES

This appendix contains examples of the different fault signature data files which can be generated by the NNFAF BIT simulator. There are four levels of BIT fault reporting which are represented: source level, intermediate (LRU) level, global fault table (GFT) level, and composite global fault table. The composite global fault table files contain fault reports from all simulated BIT techniques. The source, LRU, and composite GFT files were used as input to the neural networks.

The examples for the G-Load/Parity approach at LRU and GFT levels show the complete file (fault signature), including the signature classification at the end of the signature. All others show a few lines from the fault signature file. In fact, most of these files were substantially larger.

18.1 Source Level Data Files

The source level data files represent simulated BIT fault reports which can be captured at their source.

18.1.1 G-Load/Parity

```
! Approach: G-Load/Parity
! Curve File: CURVES/default.gp
! Boundary: 0.900000
! Threshold: 0.500000
! Seed: 1
0       0       0.062500
200     0       0.062500
400     0       0.062500
600     0       0.062500
800     0       0.062500
1000    0       0.062500
1200    0       0.064600
1400    0       0.064600
1600    0       0.066900
1800    0       0.066900
2000    0       0.069300
2200    0       0.069300
2400    0       0.069300
2600    0       0.071700
2800    0       0.071700
3000    0       0.074400
3200    0       0.074400
3400    0       0.074400
3600    0       0.077200
3800    0       0.077200
4000    0       0.077200
4200    0       0.080100
4400    0       0.080100
4600    0       0.083200
4800    0       0.083200
```

18.1.2 Vibration/Parity

```
! Approach: Vibration/Parity
! Curve File: CURVES/default.vp
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 24
0       1       0.601100
20      0       0.060200
40      1       0.375700
60      1       0.834900
80      1       0.742500
100     1       0.414400
120     0       0.259300
140     0       0.297900
160     1       0.586600
180     1       0.867400
200     1       0.592200
220     0       0.095700
240     0       0.277100
260     1       0.859700
280     1       0.825500
300     1       0.305600
320     0       0.164300
340     1       0.436800
360     1       0.671800
380     1       0.751500
400     1       0.575400
420     0       0.201600
440     0       0.232900
460     1       0.770000
480     1       0.893800
```

18.1.3 Temperature/Activity Detector

```
! Approach: Temperature/Activity Detect
! Curve File: CURVES/default.ta
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 35
3980            0       0.072500
7980            0       0.072500
11980           0       0.072500
15980           0       0.072500
19980           0       0.072500
23980           0       0.072500
27980           0       0.072500
31980           0       0.072500
35980           0       0.072500
39980           0       0.072500
43980           0       0.072500
```

47980	0	0.072500
51980	0	0.072500
55980	1	0.072500
59980	0	0.072500
63980	0	0.083700
67980	0	0.083700
71980	0	0.083700
75980	0	0.083700
79980	0	0.083700
83980	1	0.083700
87980	0	0.083700
91980	0	0.112600
95980	0	0.112600

18.1.4 Temperature/Viterbi

! Approach: Temperature/Viterbi
! Curve File: CURVES/default.tv
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 42

7980	0	0	0.072500
15980	0	0	0.072500
23980	0	0	0.072500
31980	0	0	0.072500
39980	0	0	0.072500
47980	0	0	0.072500
55980	0	0	0.072500
63980	0	0	0.083700
71980	0	0	0.083700
79980	0	0	0.083700
87980	0	0	0.083700
95980	0	0	0.112600
103980	0	0	0.112600
111980	0	0	0.112600
119980	0	0	0.112600
127980	0	0	0.156600
135980	0	0	0.156600
143980	0	0	0.156600
151980	0	0	0.212700
159980	0	0	0.212700
167980	0	0	0.212700
175980	0	0	0.212700
183980	0	0	0.277100
191980	0	0	0.277100
199980	0	0	0.277100
207980	0	0	0.277100
215980	21	0	0.346500
223980	23	0	0.346500
231980	24	0	0.346500
239980	23	0	0.346500

18.2 Intermediate (LRU) Status Level Data Files

The intermediate (LRU) level files represent the BIT fault reports which would be accessible at the LRU level. A certain amount of fault report compression might have occurred at this level. The default reporting cycle at the LRU level was simulated to be 1 second.

18.2.1 G-Load/Parity

```
! Approach: G-Load/Parity
! Curve File: CURVES/default.gp
! Boundary: 0.900000
! Threshold: 0.500000
! Seed: 1
0           0     0.062500
1000        0     0.062500
2000        0     0.069300
3000        0     0.074400
4000        0     0.077200
5000        0     0.086500
6000        0     0.093700
7000        0     0.097700
8000        0     0.111100
9000        0     0.121700
10000       0     0.127600
11000       0     0.147900
12000       0     0.164400
13000       0     0.173600
14000       0     0.206600
15000       0     0.234100
16000       0     0.250000
17000       0     0.308600
18000       0     0.360000
19000       0     0.390600
20000       0     0.510200
21000       0     0.623300
22000       0     0.694400
23000       1     1.000000
24000       1     1.000000
25000       1     1.000000
26000       1     1.000000
27000       1     1.000000
28000       1     1.000000
29000       1     1.000000
30000       1     1.000000
31000       1     1.000000
32000       1     1.000000
33000       1     1.000000
34000       1     1.000000
35000       1     1.000000
36000       1     1.000000
37000       1     1.000000
```

```
38000       1       1.000000
39000       0       0.778500
40000       1       0.694400
41000       0       0.510200
42000       0       0.425300
43000       0       0.390600
44000       0       0.308600
45000       0       0.267500
46000       0       0.250000
47000       0       0.206600
48000       0       0.183700
49000       0       0.173600
50000       0       0.147900
51000       0       0.133800
52000       0       0.127600
53000       0       0.111100
54000       0       0.101900
55000       0       0.097700
56000       0       0.086500
57000       0       0.080100
58000       0       0.077200
59000       0       0.069300
-3      0       0
```

18.2.2 Vibration/Parity

```
! Approach: Vibration/Parity
! Curve File: CURVES/default.vp
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 24
0               0       0.601100
1000            0       0.214500
-3      0       0
```

18.2.3 Temperature/Activity Detector

```
! Approach: Temperature/Activity Detect
! Curve File: CURVES/default.ta
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 35
3980    0   0.072500
7980    0   0.072500
11980   0   0.072500
15980   0   0.072500
19980   0   0.072500
23980   0   0.072500
27980   0   0.072500
31980   0   0.072500
35980   0   0.072500
```

```
39980    0    0.072500
43980    0    0.072500
47980    0    0.072500
51980    0    0.072500
55980    1    0.072500
59980    0    0.072500
63980    0    0.083700
67980    0    0.083700
71980    0    0.083700
75980    0    0.083700
79980    0    0.083700
83980    1    0.083700
87980    0    0.083700
91980    0    0.112600
95980    0    0.112600
99980    0    0.112600
```

18.2.4 Temperature/Viterbi

```
! Approach: Temperature/Viterbi
! Curve File: CURVES/default.tv
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 42
7980     0  0     0.072500
15980    0  0     0.072500
23980    0  0     0.072500
31980    0  0     0.072500
39980    0  0     0.072500
47980    0  0     0.072500
55980    0  0     0.072500
63980    0  0     0.083700
71980    0  0     0.083700
79980    0  0     0.083700
87980    0  0     0.083700
95980    0  0     0.112600
103980   0  0     0.112600
111980   0  0     0.112600
119980   0  0     0.112600
127980   0  0     0.156600
135980   0  0     0.156600
143980   0  0     0.156600
151980   0  0     0.212700
159980   0  0     0.212700
167980   0  0     0.212700
175980   0  0     0.212700
183980   0  0     0.277100
191980   0  0     0.277100
199980   0  0     0.277100
207980   0  0     0.277100
215980   26 0     0.346500
```

```
223980    31 0      0.346500
231980    29 0      0.346500
```

18.3 Global Fault Table (GFT) Status Level Data Files

The GFT level fault reports are significantly compressed by 'oring' of the original fault reports over the GFT reporting cycle. The default GFT reporting cycle was simulated to be 5 seconds.

18.3.1 G-Load/Parity

```
! Approach: G-Load/Parity
! Curve File: CURVES/default.gp
! Boundary: 0.900000
! Threshold: 0.500000
! Seed: 1
0               0       0.062500
5000            0       0.086500
10000           0       0.127600
15000           0       0.234100
20000           0       0.510200
25000           1       1.000000
30000           1       1.000000
35000           1       1.000000
40000           1       0.694400
45000           0       0.267500
50000           0       0.147900
55000           0       0.097700
-3      0       0
```

18.3.2 Vibration/Parity

```
! Approach: Vibration/Parity
! Curve File: CURVES/default.vp
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 24
0       1       0.601100
-3      0       0
```

18.3.3 Temperature/Activity Detector

```
! Approach: Temperature/Activity Detect
! Curve File: CURVES/default.ta
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 35
15000           0       0.072500
35000           0       0.072500
55000           1       0.072500
75000           0       0.083700
95000           1       0.112600
```

115000	0	0.112600
135000	0	0.156600
155000	0	0.212700
175000	0	0.212700
195000	0	0.277100
215000	0	0.346500
235000	0	0.346500
255000	2	0.417900
275000	1	0.488900
295000	3	0.488900
315000	4	0.557700
335000	4	0.623000
355000	4	0.623000
375000	4	0.683700
395000	4	0.739300
415000	4	0.739300
435000	4	0.789100
455000	4	0.832900
475000	4	0.832900
495000	4	0.870500

18.3.4 Temperature/Viterbi

```
! Approach: Temperature/Viterbi
! Curve File: CURVES/default.tv
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 42
```

15000	0	0	0.072500
35000	0	0	0.072500
55000	0	0	0.072500
75000	0	0	0.083700
95000	0	0	0.112600
115000	0	0	0.112600
135000	0	0	0.156600
155000	0	0	0.212700
175000	0	0	0.212700
195000	0	0	0.277100
215000	1	0	0.346500
235000	4	0	0.346500
255000	4	0	0.417900
275000	4	0	0.488900
295000	4	0	0.488900
315000	4	0	0.557700
335000	4	0	0.623000
355000	4	0	0.623000
375000	4	0	0.683700
395000	4	0	0.739300
415000	4	1	0.739300
435000	4	0	0.789100
455000	4	1	0.832900

```
475000      4     0          0.832900
495000      4     0          0.870500
```

18.4 Composite GFT Status Level Data Files

The composite GFT files contain the same level of BIT fault reporting as the GFT files. They also contain fault reports for all of the simulated LRUs.

18.4.1 G-Load/Parity

```
! Approach: G-Load/Parity
! Curve File: CURVES/default.gp
! Boundary: 0.900000
! Threshold: 0.500000
! Seed: 1
0           0     0     0     0     0     0.062500
5000        0     0     0     0     0     0.086500
10000       0     0     0     0     0     0.127600
15000       0     0     0     0     0     0.234100
20000       0     0     0     0     0     0.510200
25000       0     0     0     1     1     1.000000
30000       0     0     0     1     1     1.000000
35000       0     0     0     1     1     1.000000
40000       0     0     0     1     1     0.694400
45000       0     0     0     0     0     0.267500
50000       0     0     0     0     0     0.147900
55000       0     0     0     0     0     0.097700
-3    0     0
```

18.4.2 Vibration/Parity

```
! Approach: Vibration/Parity
! Curve File: CURVES/default.vp
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 24
0     0     0     0     0     1     0.601100
-3    0     0
```

18.4.3 Temperature/Activity Detector

```
! Approach: Temperature/Activity Detect
! Curve File: CURVES/default.ta
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 35
15000       0     0     0     0     0     0.072500
35000       0     0     0     0     0     0.072500
55000       1     1     1     0     0     0.072500
75000       0     0     0     0     0     0.083700
```

95000	1	1	1	0	0	0.112600
115000	0	0	0	0	0	0.112600
135000	0	0	0	0	0	0.156600
155000	0	0	0	0	0	0.212700
175000	0	0	0	0	0	0.212700
195000	0	0	0	0	0	0.277100
215000	0	0	0	0	0	0.346500
235000	0	0	0	0	0	0.346500
255000	2	2	2	0	0	0.417900
275000	1	1	1	0	0	0.488900
295000	3	3	3	0	0	0.488900
315000	4	4	4	0	0	0.557700
335000	4	4	4	0	0	0.623000
355000	4	4	4	0	0	0.623000
375000	4	4	4	0	0	0.683700
395000	4	4	4	0	0	0.739300
415000	4	4	4	0	0	0.739300
435000	4	4	4	0	0	0.789100
455000	4	4	4	0	0	0.832900
475000	4	4	4	0	0	0.832900
495000	4	4	4	0	0	0.870500

18.4.4 Temperature/Viterbi

```
! Approach: Temperature/Viterbi
! Curve File: CURVES/default.tv
! Boundary: 0.900000
! Threshold: 0.300000
! Seed: 42
```

15000	0	0	0	0	0	0.072500
35000	0	0	0	0	0	0.072500
55000	0	0	0	0	0	0.072500
75000	0	0	0	0	0	0.083700
95000	0	0	0	0	0	0.112600
115000	0	0	0	0	0	0.112600
135000	0	0	0	0	0	0.156600
155000	0	0	0	0	0	0.212700
175000	0	0	0	0	0	0.212700
195000	0	0	0	0	0	0.277100
215000	1	0	0	0	0	0.346500
235000	4	0	0	0	0	0.346500
255000	4	0	0	0	0	0.417900
275000	4	0	0	0	0	0.488900
295000	4	0	0	0	0	0.488900
315000	4	0	0	0	0	0.557700
335000	4	0	0	0	0	0.623000
355000	4	0	0	0	0	0.623000
375000	4	0	0	0	0	0.683700
395000	4	0	0	0	0	0.739300
415000	4	1	1	0	0	0.739300
435000	4	0	0	0	0	0.789100

455000	4	1	1	0	0	0.832900
475000	4	0	0	0	0	0.832900
495000	4	0	0	0	0	0.870500

19. APPENDIX H. IMPACT STUDY COST/BENEFIT DISPLAY SPREADSHEET

This appendix contains an example of the Cost/Benefit Display Spreadsheet which was developed to present the cost/benefit tradeoffs of implementing neural network false alarm filtering technology in communications systems of differing characteristics. The spreadsheet below was prepared to illustrate the types of input parameters and the resulting display. The example shows the cost/benefit analysis for implementing REINFORCE neural network false alarm filtering in a mature system.

PARAMETERS

System Name:	Example System	
System Maturity:	Mature	
NN Type:	REINFORCE	
Number of Systems:	450	
MTBF:	425	
Remaining Years of Service:	10	
Mission Duty Cycle:	0.60	
Cost Initial FAR:	20.0%	
Plotting Initial FAR:	20.0%	
Cost per repair action($K):	5	From Benefit Sheet
Avg cost per Abort($K):	20	From Benefit Sheet
Other Fixed Savings($K):	9	From Benefit Sheet
K1:	1	From Cost Sheet
K2:	0.7	From Cost Sheet

	Total Project			Per System		
FAR	Savings	NN Costs	Net	Savings	NN Costs	Net
20.00%	$9	$37,920	($37,911)	$0	$84	($84)
19.00%	$10,100	$39,361	($29,261)	$22	$87	($65)
18.00%	$19,945	$40,953	($21,008)	$44	$91	($47)
17.00%	$29,553	$42,723	($13,170)	$66	$95	($29)
16.00%	$38,932	$44,702	($5,770)	$87	$99	($13)
15.00%	$48,090	$46,932	$1,159	$107	$104	$3
14.00%	$58,836	$49,463	$9,373	$131	$110	$21
13.00%	$67,594	$52,364	$15,231	$150	$116	$34
12.00%	$76,154	$55,723	$20,431	$169	$124	$45
11.00%	$84,522	$59,663	$24,860	$188	$133	$55
10.00%	$92,705	$64,351	$28,354	$206	$143	$63
9.00%	$100,708	$70,029	$30,678	$224	$156	$68
8.00%	$108,537	$77,058	$31,479	$241	$171	$70
7.00%	$116,198	$85,999	$30,199	$258	$191	$67
6.00%	$123,697	$97,778	$25,919	$275	$217	$58
5.00%	$131,039	$114,047	$16,992	$291	$253	$38
4.00%	$138,227	$138,070	$158	$307	$307	$0
3.00%	$145,268	$177,361	($32,093)	$323	$394	($71)
2.00%	$152,166	$254,080	($101,914)	$338	$565	($226)

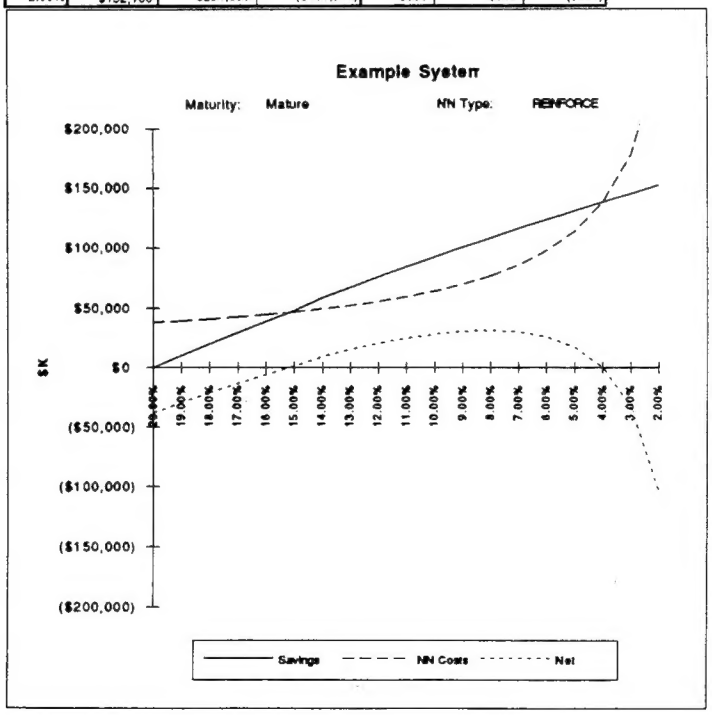


Figure H-1. Example of Cost/Benefit Display Spreadsheet

☆U.S. GOVERNMENT PRINTING OFFICE: 1995-610-126-50114

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.